# Beanstalk: A Decentralized Credit Based Stablecoin Protocol


Publius

beanstalk.publius@protonmail.com



Published: August 6, 2021

Modified: August 6, 2021

Whitepaper Version: 1.0.0

Code Version: 1.0.0


*"A national debt if it is not excessive will be to us a national blessing; it will be powerfull cement of our union."*

- Alexander Hamilton, Letter to Robert Morris, April 30, 1781[1]

## Abstract


Financial applications built on decentralized, permissionless computer networks, collectively referred to as Decentralized Finance (DeFi), often require a "stablecoin": a network-native asset with stable value relative to an arbitrary value peg (*e.g.*, 1 US Dollar (USD, $), 100 Satoshis and 1oz of Gold). To date, flawed stablecoin implementations sacrifice the main benefits of decentralized computing by requiring trust in a centralized party and limit their potential market capitalization by imposing collateral requirements. A stablecoin that (1) does not compromise on decentralization, (2) does not require collateral, and (3) trends toward more liquidity and stability, will unlock the potential of DeFi. We propose an Ethereum[2]-native, credit based stablecoin protocol that issues an ERC-20 Standard[3] token that fulfills these requirements. An on-chain price oracle leverages an existing centralized bridge between the Ethereum blockchain and the rest of the world to create a decentralized, reliable and inexpensive source for the price of a non-Ethereum-native value peg. A Decentralized Autonomous Organization (DAO) governed by a yield generating, inflationary, ERC-20 Standard token simultaneously provides security, encourages consistent liquidity growth, and dampens price volatility. Beanstalk uses a decentralized credit facility, a variable supply and a self-adjusting interest rate, to regularly cross the stablecoin price over its value peg without affecting users.


1 founders.archives.gov/documents/Hamilton/01-02-02-1167
2 ethereum.org
3 ethereum.org/en/developers/docs/standards/tokens/erc-20

## Table of Contents

# 1 Introduction

Decentralized computer networks that run on open source, permissionless protocols (*e.g.*, Bitcoin[4] and Ethereum) present the next economic and technological frontiers: trustless goods and services. Instead of requiring users to trust a rent-seeking third party to write secure code, run it on secure computer servers and perform fair system administration, trustless technology brings control back to users. Anyone can verify the security, authenticity and policies of open source software for themselves. Any computer with an internet connection can use, and participate in maintenance of, permissionless networks. Protocol-native financial incentives encourage participation in network maintenance. A diverse set of network maintenance participants creates decentralization. Decentralization and permissionlessness create censorship resistance, which is fundamental to trustlessness. Potential applications built on top of well designed trustless networks are infinite.

A key promise of decentralized, permissionless computer networks is the widespread availability of financial tools without the need for trust-providing, rent-seeking central authorities. However, as blockchains that support decentralized networks are adopted, the fiat-denominated values of their native assets (*e.g.*, Bitcoin and Ether (ETH)) change radically. To date, the practicality of using DeFi technologies is limited by the lack of a decentralized, rent-free, low-volatility blockchain-native asset.

A stablecoin protocol generates a blockchain-native asset and attempts to keep its price stable relative to an arbitrary value peg. Stablecoin utility is a function of censorship resistance, liquidity and stability. Current implementations fail to deliver a stablecoin that is (1) decentralized, (2) without collateral requirements, and (3) stable relative to its value peg.

First generation stablecoins (*e.g.*, US Dollar Coin[5] (USDC), Tether[6] and Wrapped Bitcoin[7]) claim to be 100% collateralized and require a centralized custodian. First generation stablecoins that offer convertibility function as strong bridges between their respective networks and the rest of the world. Arbitrage opportunities created by convertibility ensure the price of the on-chain asset closely tracks the custodied value peg. However, first generation stablecoins sacrifice decentralization entirely; centralized organizations custody the off-chain assets and can freeze the on-chain assets unilaterally.

Second generation stablecoins (*e.g.*, Dai[8]) use on-chain collateral to remove most points of centralization but by necessity introduce rent payments to help keep their price equal to their value peg. The combination of collateral requirements and rent payments significantly limits the potential market capitalizations of these stablecoins.

Despite the shortcomings of current stablecoin implementations, demand for USD stablecoins continues to increase rapidly. Over the past twelve months, the total market capitalization of USD stablecoins has increased more than 500% to over $100 Billion.[9] Despite this rapid increase in supply, the borrowing rates on USD stablecoins remain high,[10] indicating excess demand. Supply cannot meet market demand because of collateral requirements.

A new set of protocols have attempted to create a third generation stablecoin without collateral requirements. In general, these protocols adjust themselves mechanically to return the price of their stablecoin to their value peg. To date, implementations of third generation stablecoins have failed to regularly cross their price over their value peg due to poorly designed peg maintenance mechanisms.

[4] bitcoin.org
[5] circle.com/usdc
[6] tether.to
[7] wbtc.network
[8] makerdao.com
[9] stablecoinindex.com/marketcap
[10] app.aave.com/markets

Beanstalk uses a dynamic peg maintenance mechanism to regularly cross the price of 1 Bean (∅) – the Beanstalk ERC-20 Standard stablecoin – over its value peg without centralization or collateral requirements. Regularly crossing the price of ∅1 over its value peg creates the opportunity to regularly buy and sell Beans at its value peg. It is impossible to keep a stablecoin price equal to its value peg without convertibility; convertibility inherently introduces a centralized custodian for off-chain assets. Instead of holding a perfect peg, Beanstalk creates user confidence by consistently crossing the price of ∅1 over its value peg with increased frequency and less volatility.

Beanstalk consists of three interconnected components: (1) a decentralized price oracle, (2) a decentralized governance mechanism, and (3) a decentralized credit facility. Beanstalk-native financial incentives coordinate the components to regularly cross the price of ∅1 over its value peg during both long run decreases and increases in demand for Beans in a cost-efficient and decentralized fashion.

Beanstalk is designed from economic first principles to create a censorship resistant stablecoin. Over time, censorship resistance, liquidity and stability increase. The following principles inspire Beanstalk:

- Low concentration of ownership;
- Strong credit;
- The marginal rate of substitution;
- Low friction;
- Equilibrium; and
- Incentive structures determine behaviors of financially motivated actors.

# 2 Previous Work

Beanstalk is the culmination of previous development, evolution and experimentation within the DeFi ecosystem.

A robust trustless computer network that supports fungible token standards like Ethereum with a network-native decentralized exchange like Uniswap[11] is required to implement a decentralized stablecoin.

First generation stablecoins that offer convertibility reliably bridge the value of non-Ethereum-native assets to the Ethereum blockchain. Beanstalk leverages the existence of a centralized convertible stablecoin that trades on Uniswap to create a new decentralized stablecoin with a non-Ethereum-native value peg.

Beanstalk is inspired by Empty Set Dollar.[12] The failures of Empty Set Dollar and similar stablecoin implementations provided invaluable information that influenced the design of Beanstalk.

# 3 Decentralized Price Oracle

One problem native to decentralized stablecoin protocols is the need to be aware of a price without trusting a third-party to provide it. An oracle delivers external information to smart contracts. A robust decentralized stablecoin requires a tamper-proof, manipulation resistant and decentralized price oracle.

[11] uniswap.org
[12] emptyset.finance

When a price source is not on the blockchain, decentralized price oracles are complicated to build, expensive to maintain and often inaccurate. Beanstalk leverages Uniswap, an on-chain decentralized exchanges, and a centralized convertible stablecoin to remove these complications, costs and inaccuracies entirely.

Uniswap is an Ethereum-native decentralized exchange protocol that allows anyone to create new trading pairs between any two ERC-20 Standard tokens. Uniswap always offers a price on any size trade, at any time, for a 0.3% trading fee. Uniswap allows continuous trading in either direction by maintaining a liquidity pool of both currencies. The current price is the ratio of the two assets in the pool. Owners of both currencies can add liquidity to the pool in exchange for Uniswap liquidity pool tokens (LP tokens) unique to that liquidity pool. LP token owners receive a portion of trading fees. Price slippage is proportional to the size of a trade relative to the size of the liquidity pool. Uniswap pairs with larger liquidity pools serve as more robust price sources.

The initial Beanstalk pegs to the value of $1 by using two Uniswap v2 liquidity pools as price sources: (1) the USDC:ETH Uniswap v2 liquidity pool, which consistently has over $100 Million in liquidity,[13] and (2) a new ∅:ETH Uniswap v2 liquidity pool. Arbitrage opportunities between the USDC:ETH Uniswap pool and other exchanges ensure the pool serves as an accurate price source for USDC. Arbitrage opportunities provided by the convertibility offered between USDC and US Dollars ensure the value of 1 USDC closely tracks the value of $1. Therefore, Beanstalk considers the price of ∅1 equal to $1 when the ratios of the USDC:ETH and ∅:ETH pools are identical. The centralized organizations that control USDC cannot blacklist the USDC:ETH Uniswap pool without destroying the value proposition of USDC.

In general, a unique decentralized Beanstalk can be deployed with a value peg ($V$) for ∅1 equal to any asset (*e.g.*, $1) with an existing ERC-20 Standard first generation stablecoin ($X$) (*e.g.*, USDC) that offers convertibility to $V$ and trades on Uniswap. Beanstalk uses two Uniswap liquidity pools as price sources to construct a decentralized price oracle: (1) an existing Uniswap liquidity pool ($X$:$Y$) (*e.g.*, USDC:ETH) that consists of $X$ and a decentralized ERC-20 Standard token ($Y$) (*e.g.*, ETH[14]), and (2) a new Uniswap liquidity pool (∅:$Y$) that consists of Beans and $Y$. The combination of arbitrage opportunities between Uniswap and other exchanges, and between $X$ and $V$, ensures the $X$:$Y$ Uniswap price mirrors the exchange rate between $V$ and $Y$. Beanstalk considers the price of ∅1 equal to its value peg when the ratios of $X$:$Y$ and ∅:$Y$ are equal.

Decentralized systems are never administered by or dependent on a single individual or centralized organization. Beanstalk can leverage an arbitrary $X$ without exposure to malicious actions from its centralized operators (*e.g.*, censorship) by maintaining an identical ratio of $X$:$Y$ and ∅:$Y$.

To increase the cost of oracle manipulation, instead of using the last traded price, the Beanstalk price oracle calculates a time weighted average price[15] (TWAP) for each pair ($\bar{P}^{X:Y}$ and $\bar{P}^{\emptyset:Y}$), such that $\bar{P}^{X:Y},\ \bar{P}^{\emptyset:Y} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, by averaging the last traded price in each Ethereum block over a predefined time interval.

We define the oracle price of ∅1 ($\bar{P}$), such that $\bar{P} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, for a given $\bar{P}^{X:Y}$ and $\bar{P}^{\emptyset:Y}$ as:

$$\bar{P} = \frac{\bar{P}^{\emptyset:Y}}{\bar{P}^{X:Y}}$$

When $\bar{P} = 1$, the TWAP of ∅1 was equivalent to $V$ over the time interval. Thus, Beanstalk constructs a robust cost-efficient decentralized price oracle for a non-Ethereum-native value peg.

[13] v2.info.uniswap.org/pair/0xb4e16d0168e52d35cacd2c6185b44281ec28c9dc
[14] weth.io
[15] uniswap.org/docs/v2/core-concepts/oracles

# 4 The Bean Farm

Well designed decentralized protocols create utility for end users without requiring, but never limiting, participation in protocol maintenance. Protocol-native financial incentives encourage performance of work that creates utility for end users. Low barriers to and variety in work enable a diverse set of participants. A diverse set of well incentivized workers creates censorship resistant utility.

Beanstalk does not require actions from, impose rent on, or affect in any way, regular Bean users (*e.g.*, smart contracts). Anyone with ETH and Beans or LP tokens for the ∅:$Y$ Uniswap liquidity pool ($\Lambda$) can join the *Bean Farm* and profit from participation in protocol maintenance. Governance of Beanstalk upgrades and Bean peg maintenance take place on the *Bean Farm*.

The *Bean Farm* has two components: the *Silo* and *Field*. Beanstalk-native financial incentives coordinate the two components to create a stalwart system of governance, consistently grow Bean liquidity and regularly cross the price of ∅1 over its value peg.

The *Silo* – the Beanstalk DAO – offers passive yield opportunities to Bean and $\Lambda$ owners for participation in governance of Beanstalk upgrades. Anyone can become a *Silo Member* by *Depositing* ∅ or $\Lambda$ into the *Silo* to earn *Stalk*. *Stalk* owners govern Beanstalk upgrades and are rewarded with Beans when the Bean supply increases.

The *Field* – the Beanstalk lending facility – offers yield opportunities to *Bean Farmers* (creditors) for participation in peg maintenance. Anyone can become a *Bean Farmer* by lending Beans that are not in the *Silo* to Beanstalk. Bean loans are repaid to *Bean Farmers* with interest when the Bean supply increases.

Time on the *Bean Farm* is kept in *Seasons*.

# 5 Seasons

The Beanstalk governance and peg maintenance mechanisms require a protocol-native timekeeping mechanism and regular code execution on the Ethereum blockchain. Beanstalk uses *Seasons* to create a cost-efficient protocol-native timekeeping mechanism and to ensure cost-efficient code execution on Ethereum at regular intervals. In general, Beanstalk uses Ethereum block timestamps ($E$) such that $E \in \mathbb{Z}^+$.

We define a *Season* ($t$), such that $t \in \mathbb{Z}^+$, as an approximately 3,600 second (1 Hour) interval. Every *Season* begins when the `sunrise()` function is called manually on the blockchain. When Beanstalk accepts the `sunrise()` function, the necessary code is executed.

The `sunrise()` function is called by sending a transaction on the Ethereum blockchain that includes a `sunrise()` function call. Beanstalk only accepts the first `sunrise()` function call such that the timestamp in the Ethereum block containing it is sufficiently distant from the timestamp in the Ethereum block containing the Beanstalk deployment ($E_1$).

The minimum timestamp Beanstalk accepts a `sunrise()` function call for a given $t$ ($E_t^{\text{min}}$) and $E_1$ is:

$$E_t^{\text{min}} = 3600(t-1) + E_1$$

The cost to execute the code changes depending on the traffic on the Ethereum network and the state of Beanstalk. Beanstalk covers the transaction cost by awarding the sender of an accepted `sunrise()` function call with newly minted Beans. To encourage regular `sunrise()` function calls even during periods of congestion on the Ethereum network while minimizing cost, the award starts at 100 Beans and compounds 1% every additional second that elapses past $E_t^{\text{min}}$ for 300 seconds.

The award for successfully calling the `sunrise()` function for $t$ ($a_t$), such that $a_t \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$, in a block with a given timestamp ($E_t$) is:

$$a_t = 100 \times 1.01^{\min\{E_t - E_t^{\min},\ 300\}}$$

To minimize the cost of calculating $a_t$, Beanstalk uses a binomial estimation with a margin of error of less than 0.05%. Thus, Beanstalk creates a cost-efficient protocol-native timekeeping mechanism and ensures cost-efficient code execution on the Ethereum blockchain at regular intervals.

# 6 Silo

Beanstalk requires the ability to coordinate protocol upgrades. The *Silo* – the Beanstalk DAO – uses the *Stalk System* to create protocol-native financial incentives that coordinate Beanstalk upgrades and consistently improve security, liquidity and stability. *Silo Members* earn passive yield from participation in governance of Beanstalk upgrades.

## 6.1 The Stalk System

The *Stalk System* improves decentralization and creates Beanstalk-native financial incentives to leave assets *Deposited* in the *Silo* and add liquidity to the Ø:$Y$ pool.

Anyone can become a *Silo Member* by *Depositing* Ø or $\Lambda$ into the *Silo* to earn *Seeds* and *Stalk*. *Seeds* and *Stalk* are both ERC-20 Standard tokens. Every *Season*, $1 \times 10^{-4}$ additional *Stalk* grows from each *Seed*. *Silo Membership* is proportional to *Stalk* ownership relative to total outstanding *Stalk*.

*Silo Members* are entitled to participate in Beanstalk governance and a portion of Bean mints. The influence in governance of, and distribution of Beans paid to, a *Silo Member* are proportional to their *Silo Membership*. *Silo Membership* becomes less concentrated over time.

## 6.2 Deposits and Withdrawals

Beans and $\Lambda$ can be *Deposited* into or *Withdrawn* from the *Silo* at any time. Assets are *Frozen* for 24 full *Seasons* (∼1 Day) upon *Withdrawal*.

Beanstalk rewards *Seeds* and *Stalk* to *Depositors* immediately upon *Depositing* Ø or $\Lambda$ into the *Silo* based on the Bean-denominated amount of *Deposit* and asset (*i.e.*, Ø and $\Lambda$) *Deposited*. Ø *Depositors* receive 2 *Seed* and 1 *Stalk* per Bean *Deposited*. $\Lambda$ *Depositors* receive 4 *Seeds* and 1 *Stalk* per Bean *Deposited*. Beans paid to *Stalk* owners are automatically *Deposited* into the Silo and receive *Seeds* and *Stalk*.

The amount of *Seeds*, *Stalk*, and *Stalk* from *Seeds* rewarded for a *Deposited* asset must be forfeited upon *Withdrawal* from the *Silo*.

Beans that are *Deposited* in the *Silo* can be *Converted* to *Deposited* $\Lambda$ by contributing additional Y (*e.g.*, ETH) to the Ø:$Y$ Uniswap liquidity pool. No *Stalk* are forfeited when *Converting Deposited* Ø to *Deposited* $\Lambda$.

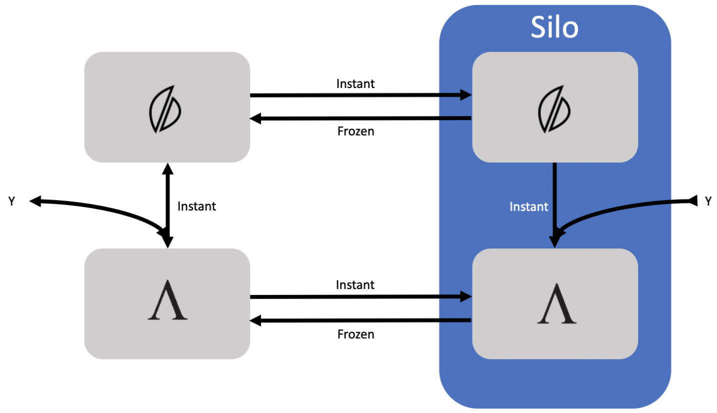


Figure 1: Silo

### 6.3 Calculating Stalk

A *Silo Member's* total *Stalk* is the sum of the *Stalk* for each of their *Deposits*. The *Stalk* for a given *Deposit* are determined by its duration of *Deposit*, asset type and Bean-denominated amount when *Deposited*.

Beanstalk stores two maps of each member's *Deposits* that are still in the *Silo*: (1) a map of $\emptyset$ *Deposited* each *Season* ($\lambda^{\emptyset}$), and (2) a map of the amounts and Bean-denominated amounts of $\Lambda$ *Deposited* each *Season* ($\lambda^{\Lambda}$).

When a *Silo Member Deposits* $\emptyset$, they update the number of $\emptyset$ *Deposited* ($z_i^{\emptyset}$), such that $z_i^{\emptyset} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, in $\lambda^{\emptyset}$, where $i$ corresponds to the *Season* of *Deposit*.

The *Seeds* for an updated $\lambda^{\emptyset}$ ($c_t^{\emptyset}$), such that $c_t^{\emptyset} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, is:

$$c_t^{\emptyset} = \sum_{z_i^{\emptyset} \in \lambda^{\emptyset}} 2z_i^{\emptyset}$$

The *Stalk* during a given $t$ for an updated $\lambda^{\emptyset}$ ($k_t^{\emptyset}$), such that $k_t^{\emptyset} \in \{j \times 10^{-10} \mid j \in \mathbb{N}\}$, is:

$$k_t^{\emptyset} = \sum_{z_i^{\emptyset} \in \lambda^{\emptyset}} z_i^{\emptyset}(1 + \frac{(t-i)}{5000})$$

When a *Silo Member Deposits* $\Lambda$, they update the amount of $\Lambda$ *Deposited* ($z_i^{\Lambda}$) and the Bean-denominated amount of $\Lambda$ *Deposited* ($z_i^{\Lambda:\emptyset}$), such that $z_i^{\Lambda},\ z_i^{\Lambda:\emptyset} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, in $\lambda^{\Lambda}$, where $i$ corresponds to the *Season* of *Deposit*.

For a given list of $\Lambda$ *Deposits* ($h$) during $i$ ($l_i^{\Lambda}$), $z_i^{\Lambda:\emptyset}$ is calculated from the number of Beans in the $\emptyset$:$Y$ liquidity pool at the time of each *Deposit* ($b_h$), such that $b_h \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, and the total $\Lambda$ at the time of each *Deposit* ($\Lambda_h$), such that $h,\ \Lambda_h \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, as:

$$z_i^{\Lambda:\emptyset} = \sum_{h \in l_i^{\Lambda}} \frac{2b_h \times h}{\Lambda_h}$$

The *Seeds* for an updated $\lambda^{\Lambda}$ ($c_t^{\Lambda}$), such that $c_t^{\Lambda} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, is:

$$c_t^{\Lambda} = \sum_{z_i^{\Lambda:\emptyset} \in \lambda^{\Lambda}} 4z_i^{\Lambda:\emptyset}$$

The *Stalk* during a given $t$ for an updated $\lambda^{\Lambda}$ ($k_t^{\Lambda}$), such that $k_t^{\Lambda} \in \{j \times 10^{-10} \mid j \in \mathbb{N}\}$, is:

$$k_t^{\Lambda} = \sum_{z_i^{\Lambda:\emptyset} \in \lambda^{\Lambda}} z_i^{\Lambda:\emptyset}(1 + \frac{(t-i)}{2500})$$

A *Silo Member's* total *Seeds* ($C_t$), such that $C_t \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, is:

$$C_t = c_t^{\emptyset} + c_t^{\Lambda}$$

A *Silo Member's* total *Stalk* during a given $t$ ($K_t$), such that $K_t \in \{j \times 10^{-10} \mid j \in \mathbb{Z}^{+}\}$, is:

$$K_t = k_t^{\emptyset} + k_t^{\Lambda}$$

When a *Silo Member Withdraws* Beans or $\Lambda$, they must forfeit the amount of *Seeds* and *Stalk* rewarded to the assets being *Withdrawn* and update the appropriate map accordingly. When a *Silo Member Converts Deposited* $\emptyset$ to *Deposited* $\Lambda$, they transfer the corresponding entries from $\lambda^{\emptyset}$ to $\lambda^{\Lambda}$ and update the *Seasons* of *Deposit* to retain their rewarded *Seeds* and *Stalk*.

## 6.4 Governance

A robust decentralized governance mechanism must balance the principles of decentralization with resistance to attempted protocol changes, both malicious and ignorant, and the ability to quickly adapt to changing information. In practice, Beanstalk must balance ensuring sufficient time for all ecosystem participants to consider a *BIP*, join the *Silo* and cast their votes, with the ability to quickly upgrade in cases of emergency.

### 6.4.1 Participation

Any Bean or $\Lambda$ owner can become a *Silo Member* and participate in Beanstalk governance by *Depositing* $\emptyset$ or $\Lambda$ into the *Silo* to earn *Stalk*.

Any *Silo Member* that owns more than 0.1% of total outstanding *Stalk* can submit a *BIP*. In the future, as the ownership concentration of *Stalk* decreases, we expect a *BIP* to lower this threshold.

The award for submitting a *BIP* that gets accepted ($a^{BIP}$), such that $a^{BIP} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, is determined by the author of the *BIP*. If $a^{BIP}$ is excessively high such that a *BIP* that would otherwise be acceptable to the community is voted down because of the award, the open source nature of Beanstalk allows someone else to re-submit an identical *BIP* with a more reasonable $a^{BIP}$.

Beanstalk only accepts votes in favor of a *BIP*. A *Silo Member's* vote is counted in proportion to their *Stalk*. A *Silo Member* is unable to *Withdraw* their assets from the *Silo* after casting a vote in favor of a *BIP* until the end of the *Voting Period* or they rescind their vote. The submitter of a *BIP* automatically votes in favor of the *BIP* and they cannot rescind their vote.

### 6.4.2 Voting Period

A *Voting Period* opens when a *BIP* is submitted to the Ethereum blockchain and ends at the beginning of the 169th *Season* after it is submitted, or when it is committed with a supermajority.

If at the end of the *Voting Period*:

- Less than or equal to half of the total outstanding *Stalk* votes in favor of the $BIP$, it fails; or
- More than half of the total outstanding *Stalk* votes in favor of the $BIP$, it passes.

If at any time 24 Hours or more after the beginning and before the end of the *Voting Period* more than two-thirds of the total outstanding *Stalk* votes in favor of the $BIP$, it can be committed to the Ethereum blockchain.

#### 6.4.3 Pause

In case of a particularly dangerous vulnerability to Beanstalk, the *Silo* can *Pause* or *Unpause* Beanstalk by a two-thirds supermajority vote at any time or as part of a normal $BIP$. When *Paused*, Beanstalk does not accept a `sunrise()` function call. When *Unpaused*, the `sunrise()` function can be called at the beginning of the next hour.

For a given timestamp of last *Unpause* ($E_f$) during *Season* $t^{'}$, we define $E_t^{\min}$ $\forall$ $E_t^{\min}$ such that $t > t^{'}$ as:

$$E_t^{\min} = 3600\left(\left\lceil \frac{E_f}{3600} \right\rceil + t - t^{'} - 1\right)$$

At launch, the deployment address has the ability to *Pause* or *Unpause* Beanstalk unilaterally. In the future, we expect a $BIP$ will revoke this ability from the deployment address.

#### 6.4.4 Beanstalk Improvement Proposals

Beanstalk implements EIP-2535.[16] Beanstalk is a diamond with multiple facets. Beanstalk supports multiple simultaneous $BIP$ with independent *Voting Periods*.

A BIP has four inputs: (1) Beanstalk should *Pause* or *Unpause*, (2) a list of facets and functions to add or remove upon commit, (3) a function to run upon commit, and (4) the Ethereum address of the contract holding the function to run upon commit.

If inputs 2, 3 and 4 are empty, a $BIP$ can pass with a two-thirds supermajority vote at any time before the end of the *Voting Period*.

#### 6.4.5 Commit

When a $BIP$ passes or has a two-thirds majority, it must be manually committed to the Ethereum blockchain. To encourage prompt commitment of $BIP$ even during periods of congestion on the Ethereum network while minimizing cost, the award for successful commitment starts at 100 Beans and compounds 1% every additional five seconds that elapse past the end of its *Voting Period* ($E_{BIP}$) for 1,500 seconds.

The award for successfully committing an approved $BIP$ ($a^q$), such that $a^q \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, with a given timestamp of commitment ($E_q$) and $E_{BIP}$ is:

$$a^q = 100 \times 1.01^{\min\left\{\left\lfloor \frac{E_q - E_{BIP}}{5} \right\rfloor,\ 300\right\}}$$

To minimize the cost of calculating $a^q$, Beanstalk uses a binomial estimation with a margin of error of less than 0.05%. When a $BIP$ is passed with a two-thirds supermajority before the end of its *Voting Period*, $a^q = 1000$.

[16] eips.ethereum.org/EIPS/eip-2535

Thus, Beanstalk creates a robust decentralized governance mechanism and consistent improves security, liquidity and stability.

# 7 Field

Beanstalk is a credit based stablecoin. The *Field* is the Beanstalk lending facility.

Anytime there is *Soil* (defined below) in the *Field*, any owner of Beans that are not in the *Silo* or *Frozen* can *Sow* (lend) Beans to Beanstalk in exchange for *Pods* (defined below) and become a *Bean Farmer*. The *Weather* is the interest rate on Bean loans.

## 7.1 Soil

We define *Soil* ($S$), such that $S \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, as the current number of Beans that can be *Sown* in exchange for *Pods*. ∅1 is *Sown* in one *Soil*. Any time there is *Soil* in the *Field* (*i.e.*, $S > 0$), Beans can be *Sown* into *Soil*. Beanstalk permanently removes *Sown* ∅ from the Bean supply.

When Beanstalk is willing to remove more Beans from the Bean supply, it creates more *Soil*. Beanstalk increases the *Soil* supply at the beginning of each *Season* according to the peg maintenance mechanism.

## 7.2 Pods

*Pods* are the debt asset of Beanstalk. Beanstalk never defaults on debt: *Pods* automatically grow from *Sown* ∅ and never expire.

In the future, when the TWAP of ∅1 is above its value peg over a *Season*, *Pods* ripen and become *Harvestable* (redeemable) for ∅1 at anytime. *Pods* ripen on a first in, first out (FIFO) basis: Beans that are *Sown* first ripen into *Harvestable Pods* first. *Bean Farmers* can *Harvest* their ripened *Pods* anytime by submitting a `harvest()` function call to the blockchain. There is no penalty for waiting to *Harvest Pods*.

*Pods* are transferable. In practice, *Pods* are non-callable bonds with priority for payment represented as a place in line. The number of *Pods* that grow from *Sown* ∅ is determined by the *Weather*.

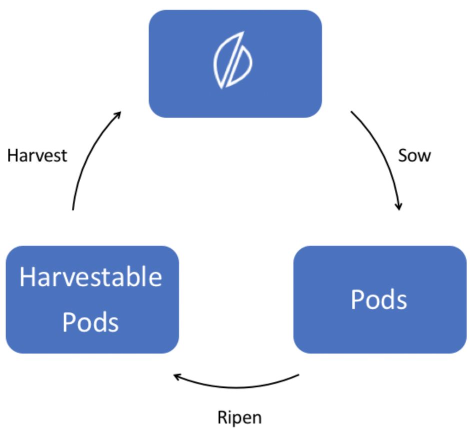


Figure 2: Field

### 7.3 Weather

We define the *Weather* ($w$), such that $w \in \mathbb{Z}^+$, as the percentage of additional Beans ultimately *Harvested* from 1 *Sown* Ø.

The number of *Pods* ($d$) that grow from a given number of *Sown* Ø ($u$), such that $d,\ u \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, and $w$ is:

$$d = u \times \left(1 + \frac{w}{100}\right)$$

The *Weather* is constant each *Season*. Beanstalk changes the *Weather* at the beginning of each *Season* according to the peg maintenance mechanism.

# 8 Peg Maintenance

Beanstalk faces the fundamental limitation that it cannot fix the price of Ø1 at its value peg, but instead must encourage widespread participation in peg maintenance through protocol-native financial incentives. Stability is a function of how frequently and regularly the price of Ø1 crosses, and the magnitudes of price deviations from, its value peg. Beanstalk regularly crosses the price of Ø1 over its value peg during both long run decreases and increases in demand for Beans.

Beanstalk has four optional peg maintenance tools available: (1) increase the Bean supply, (2) increase the *Soil* supply, (3) change the *Weather*, and (4) sell Beans on Uniswap. At the beginning of every *Season*, Beanstalk evaluates its position (*i.e.*, price and debt level) and current state (*i.e.*, direction and acceleration) with respect to ideal equilibrium (defined below), and dynamically adjusts the Bean supply, *Soil* supply and *Weather* to move closer to ideal equilibrium.

### 8.1 Ideal Equilibrium

Beanstalk is credit based. Beanstalk only fails if it can no longer attract creditors. A reasonable level of debt attracts creditors. Therefore, in addition to the Bean price, the peg maintenance mechanism considers the Beanstalk debt level (defined below).

Beanstalk is in ideal equilibrium when the Bean price and the Beanstalk debt level are both stable at their optimal levels. In practice, this requires that three conditions are met: (1) the price of Ø1 is regularly oscillating over its value peg, (2) the Beanstalk debt level is optimal (defined below), and (3) demand for *Soil* is steady (defined below).

Beanstalk affects the supply of and demand for Beans to return to ideal equilibrium in response to the Bean price, the Beanstalk debt level and changing demand for *Soil* by adjusting the Bean supply, *Soil* supply, and *Weather*. Increases to the Bean or *Soil* supply primarily affect Bean supply. Changes to the *Weather* primarily affect demand for Beans. In order to make the proper adjustments, Beanstalk closely monitors the states of both the Bean and *Soil* markets.

In practice, maintaining ideal equilibrium is impossible. Deviations from ideal equilibrium along one or both axes are normal and expected. As Beanstalk grows, the durations and magnitudes of deviations decrease.

## 8.2 Bean Supply

At the beginning of each *Season*, independently of the award for successfully calling the `sunrise()` function, Beanstalk increases the Bean supply by the time weighted average shortage of Beans in the ₵:$Y$ liquidity pool over the previous *Season*. Half the additional Bean supply increase is used to pay off debt and the other half is distributed to *Stalk* owners.

At the beginning of $t$, Beanstalk calculates the TWAP over the previous *Season* of ₵1 ($\bar{P}_{t-1}$), and each pool ($\bar{P}^{X:Y}_{t-1}$, $\bar{P}^{₵:Y}_{t-1}$), and a time weighted average shortage or excess of Beans in the ₵:$Y$ liquidity pool over the previous *Season* ($\Delta\bar{b}_{t-1}$), such that $\Delta\bar{b}_{t-1} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}\}$.

If $\bar{P}_{t-1} > 1$, there was a time weighted average shortage of Beans in the pool over the previous *Season* (*i.e.*, $\Delta\bar{b}_{t-1} > 0$). If $\bar{P}_{t-1} < 1$, there was a time weighted average excess of Beans in the pool over the previous *Season* (*i.e.*, $\Delta\bar{b}_{t-1} < 0$). If $\bar{P}_{t-1} = 1$, the TWAP of ₵1 was equal to its value peg over $t$.

$\Delta\bar{b}_{t-1}$ is computed from the difference between the time weighted average optimal number of Beans in the ₵:$Y$ liquidity pool over the previous *Season* ($\bar{b}^*_{t-1}$) and the time weighted average number of Beans in the ₵:$Y$ liquidity pool over the previous *Season* ($\bar{b}_{t-1}$), such that $\bar{b}^*_{t-1},\ \bar{b}_{t-1} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$.

We define $\bar{b}^*_{t-1}$ for a given $\bar{P}^{X:Y}_{t-1}$, number of ₵ in the ₵:$Y$ liquidity pool at the end of the previous *Season* ($b_{t-1}$), such that $b_{t-1} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, and number of $Y$ in the ₵:$Y$ liquidity pool at the end of the previous *Season* ($y_{t-1}$), such that $y_{t-1} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, as:

$$\bar{b}^*_{t-1} = \sqrt{\frac{b_{t-1} \times y_{t-1}}{\bar{P}^{X:Y}_{t-1}}}$$

We define $\bar{b}_{t-1}$ for a given $\bar{P}^{₵:Y}_{t-1}$, $b_{t-1}$ $y_{t-1}$ as:

$$\bar{b}_{t-1} = \sqrt{\frac{b_{t-1} \times y_{t-1}}{\bar{P}^{₵:Y}_{t-1}}}$$

Therefore, we define $\Delta\bar{b}_{t-1}$ for a given $\bar{b}^*_{t-1}$ and $\bar{b}_{t-1}$ as:

$$\Delta\bar{b}_{t-1} = \bar{b}^*_{t-1} - \bar{b}_{t-1}$$

At the beginning of each *Season*, Beanstalk mints $m_t$ Beans, such that $m_t \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$. We define $m_t$ for a given $a_t$ and $\Delta\bar{b}_{t-1}$ as:

$$m_t = a_t + \max(0,\ \Delta\bar{b}_{t-1})$$

The distribution of $\Delta\bar{b}_{t-1}$ is dependant on the total number of unripened *Pods* ($D$), such that $D \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, and $\Delta\bar{b}_{t-1}$. If $D > \frac{1}{2}\Delta\bar{b}_{t-1}$ (*i.e.*, there are more unripened *Pods* than $\frac{1}{2}\Delta\bar{b}_{t-1}$), $\frac{1}{2}\Delta\bar{b}_{t-1}$ *Pods* ripen and become *Harvestable* and $\frac{1}{2}\Delta\bar{b}_{t-1}$ newly minted Beans are distributed to *Stalk* owners. If $D < \frac{1}{2}\Delta\bar{b}_{t-1}$ (*i.e.*, there are less unripened *Pods* than $\frac{1}{2}\Delta\bar{b}_{t-1}$), $D$ *Pods* ripen and become *Harvestable* and $\Delta\bar{b}_{t-1} - D$ newly minted Beans are distributed to *Stalk* owners.

## 8.3 Soil Supply

At the beginning of each *Season* Beanstalk increases the *Soil* supply based on the time weighted average excess in the ₵:$Y$ liquidity pool over the previous *Season* and the *Soil Rate*. Beans may not always be *Sown* immediately in *Soil*. *Soil* keeps a running total of unaddressed Bean excesses in the ₵:$Y$ liquidity pool. The *Soil Rate* represents the portion of the total Bean supply Beanstalk is currently willing to remove in exchange for debt.

Beanstalk does not consider *Sown* ∅ *Burnt Beans* or unripened *Pods*, but does consider *Harvestable Pods* as part of the total Bean supply.

We define the total Bean supply ($B$) for a given total Beans minted over all *Seasons* ($M$), such that $B,\ M, \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$, total $a^{BIP}$ for all passed $BIP$ ($A^{BIP}$), total $a^{q}$ for all committed $BIP$ ($A^{q}$), total *Burnt Beans* over all *Seasons* ($G$) and total *Sown* ∅ over all *Seasons* ($U$), such that $A^{BIP},\ A^{q},\ G,\ U \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, as:

$$B = M + A^{BIP} + A^{q} - (G + U)$$

We define the *Soil Rate* ($R^{S}$), such that $R^{S} \in \{j \times 10^{-6} \mid j \in \mathbb{N},\ j \leq 10^{6}\}$, for a given $S$ and $B$ as:

$$R^{S} = \frac{S}{B}$$

Beanstalk is willing to issue debt every *Season*. To enforce this policy, Beanstalk requires a *Minimum Soil Rate* ($R^{S^{\min}}$), such that $R^{S^{\min}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+},\ j \leq 10^{6}\}$, that serves as a lower limit on *Soil* relative to the Bean supply at the start of the *Season* ($B_t$).

The *Minimum Soil* at the beginning of $t$ ($S_t^{\min}$), such that $S_t^{\min} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$, Beanstalk has outstanding for a given $R^{S^{\min}}$ and $B_t$ is:

$$S_t^{\min} = R^{S^{\min}} \times B_t$$

When there are rapid increases or decreases in marginal supply of or demand for Beans, respectively, it may take Beanstalk multiple *Seasons* to raise the *Weather* sufficiently to remove Beans from the supply. In the meantime, creating additional *Soil* does not move Beanstalk closer to ideal equilibrium but does expose Beanstalk to overpaying for *Sown* ∅. Therefore, Beanstalk requires a *Maximum Soil Rate* ($R^{S^{\max}}$), such that $R^{S^{\max}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+},\ j \leq 10^{6}\}$, that serves as an upper limit on *Soil* relative to $B_t$.

The *Maximum Soil* at the beginning of $t$ ($S_t^{\max}$), such that $S_t^{\max} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$, Beanstalk has outstanding for a given $R^{S^{\max}}$ and $B_t$ is:

$$S_t^{\max} = R^{S^{\max}} \times B_t$$

Therefore, the *Soil* supply at the beginning of each *Season* ($S_t^{\text{start}}$), such that $S_t^{\text{start}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$, for a given *Soil* supply at the end of the previous *Season* ($S_{t-1}^{\text{end}}$), such that $S_{t-1}^{\text{end}} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, $\Delta\bar{b}_{t-1}$, $S_t^{\min}$ and $S_t^{\max}$ is:

$$S_t^{\text{start}} = \min(\max(S_{t-1}^{\text{end}} - \Delta\bar{b}_{t-1},\ S_t^{\min}),\ S_t^{\max})$$

At the beginning of each *Season*, Beanstalk mints $s_t$ *Soil*, such that $s_t \in \{j \times 10^{-6} \mid j \in \mathbb{Z}\}$.

We define $s_t$ for a given $S_t^{\text{start}}$ and $S_{t-1}^{\text{end}}$ as:

$$s_t = S_t^{\text{start}} - S_{t-1}^{\text{end}}$$

## 8.4 Weather

Beanstalk regularly crosses the price of ∅1 over its value peg during long run decreases and increases in demand for Beans primarily by adjusting the *Weather* each *Season*.

At the beginning of $t$, Beanstalk changes the *Weather* depending on its debt level and current state with respect to ideal equilibrium. The current state (defined below) is a function of the position of Beanstalk with respect to ideal equilibrium and changing demand for *Soil*. The position of Beanstalk with respect to ideal equilibrium is a function of $\bar{P}_{t-1}$ and the Beanstalk debt level.

### 8.4.1 Debt Level

The *Pod Rate* ($R^D$), such that $R^D \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, represents the Beanstalk debt level relative to the Bean supply. The *Weather* change are determined in part by $R^D$.

We define the $R^D$ for a given $D$ and $B$ as:

$$R^D = \frac{D}{B}$$

Beanstalk requires three $R^D$ levels to be set: (1) $R^{D^{\text{lower}}}$, below which debt is considered excessively low, (2) $R^{D^*}$, an optimal level of debt, and (3) $R^{D^{\text{upper}}}$, above which debt is considered excessively high, such that $R^{D^{\text{lower}}}$, $R^{D^*}$, $R^{D^{\text{upper}}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$. When $R^D$ is between $R^{D^{\text{lower}}}$ and $R^{D^{\text{upper}}}$, but not optimal, $R^D$ is considered reasonable.

When $R^D$ is excessively high or low, Beanstalk changes the *Weather* more aggressively.

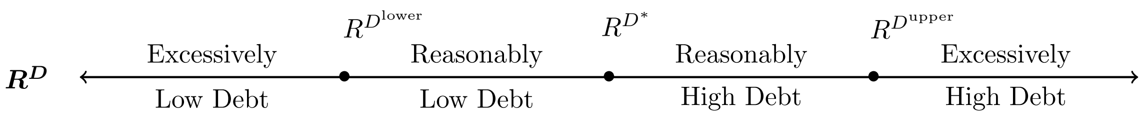


Figure 3: Debt Level

### 8.4.2 Position

The position of Beanstalk with respect to ideal equilibrium can be represented on a graph with axes $\bar{P}$ and $R^D$, and ideal equilibrium at the origin ($R^{D^*}$, 1). The current state of Beanstalk is in part determined by the position of Beanstalk with respect to ideal equilibrium.

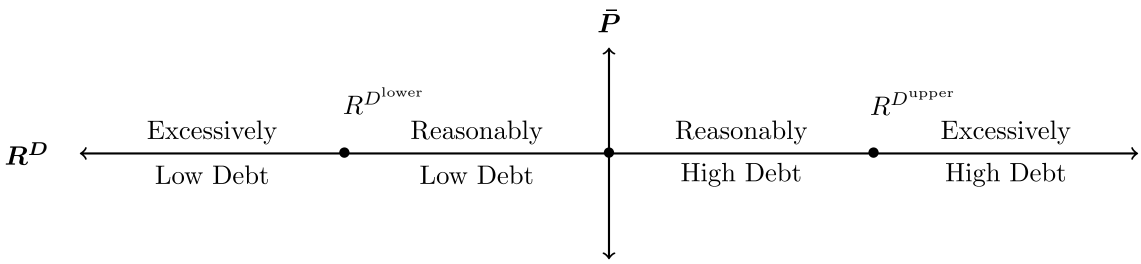


Figure 4: Position

### 8.4.3 Direction

The position of Beanstalk with respect to ideal equilibrium changes each *Season*. The current state of Beanstalk with respect to ideal equilibrium is in part determined by the direction of this change.

The direction of change in position of Beanstalk is considered either toward or away from ideal equilibrium based on the current $R^D$ and $\bar{P}_{t-1}$. When $\bar{P}_{t-1} > 1$, Beanstalk pays off debt; when $\bar{P}_{t-1} < 1$, debt can only increase.

Therefore, when $R^D > R^{D^*}$ (*i.e.*, there is more debt than optimal):

- If $\bar{P}_{t-1} > 1$, Beanstalk moves toward ideal equilibrium; or
- If $\bar{P}_{t-1} < 1$, Beanstalk moves away from ideal equilibrium.

When $R^D < R^{D^*}$ (*i.e.*, there is less debt than optimal):

- If $\bar{P}_{t-1} > 1$, Beanstalk moves away from ideal equilibrium; or
- If $\bar{P}_{t-1} < 1$, Beanstalk moves toward ideal equilibrium.

| | | $R^D$ | | | |
|---|---|---|---|---|---|
| | **Direction** | **Excessively Low Debt** | **Reasonably Low Debt** | **Reasonably High Debt** | **Excessively High Debt** |
| $\bar{P}$ | $\bar{P}_{t-1} > 1$ | Away From | Away From | Toward | Toward |
| | $\bar{P}_{t-1} < 1$ | Toward | Toward | Away From | Away From |

Figure 5: Direction

#### 8.4.4 Acceleration

The rate of change of position of Beanstalk each *Season* changes. The current state of Beanstalk with respect to ideal equilibrium is also determined by its acceleration.

The acceleration of Beanstalk is considered accelerating, steady or decelerating based on $\bar{P}_{t-1}$ and changing demand for *Soil* (defined below).

When demand for *Soil* is decreasing:

- If $\bar{P}_{t-1} > 1$, Beanstalk is decelerating; or
- If $\bar{P}_{t-1} < 1$, Beanstalk is accelerating.

When demand for *Soil* is steady, Beanstalk is steady.

When demand for *Soil* is increasing:

- If $\bar{P}_{t-1} > 1$, Beanstalk is accelerating; or
- If $\bar{P}_{t-1} < 1$, Beanstalk is decelerating.

| | | Demand | | |
|---|---|---|---|---|
| | **Acceleration** | **Decreasing Demand** | **Steady Demand** | **Increasing Demand** |
| $\bar{P}$ | $\bar{P}_{t-1} > 1$ | Decelerating | Steady | Accelerating |
| | $\bar{P}_{t-1} < 1$ | Accelerating | Steady | Decelerating |

Figure 6: Acceleration

#### 8.4.5 Demand for Soil

In order to properly classify its acceleration, Beanstalk must accurately measure changing demand for *Soil*. Demand for *Soil* is considered decreasing, steady or increasing.

The change in *Soil* from the beginning to the end of each *Season* ($\Delta S_t$), such that $\Delta S_t \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, indicates demand for *Soil* over the course of that *Season*. The rate of change of $\Delta S_t$ from *Season* to *Season* ($\frac{\partial \Delta S}{\partial t}$), such that $\frac{\partial \Delta S}{\partial t} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, indicates changing demand for *Soil*.

We define $\Delta S_t$ for a given $S_t^{\text{start}}$ and $S_t^{\text{end}}$ as:

$$\Delta S_t = S_t^{\text{start}} - S_t^{\text{end}}$$

We define $\frac{\partial \Delta S}{\partial t}$ for given $\Delta S_t$ over the previous two *Seasons*, $\Delta S_{t-1}$ and $\Delta S_{t-2}$, respectively, as:

$$\frac{\partial \Delta S}{\partial t} = \frac{\Delta S_{t-1}}{\Delta S_{t-2}}$$

Beanstalk requires two $\frac{\partial \Delta S}{\partial t}$ levels to be set: (1) $\frac{\partial \Delta S}{\partial t}^{\text{lower}}$, below which demand for *Soil* is considered decreasing, and (2) $\frac{\partial \Delta S}{\partial t}^{\text{upper}}$, above which demand for *Soil* is considered increasing, such that $\frac{\partial \Delta S}{\partial t}^{\text{lower}}, \frac{\partial \Delta S}{\partial t}^{\text{upper}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$. When $\frac{\partial \Delta S}{\partial t}$ is between $\frac{\partial \Delta S}{\partial t}^{\text{lower}}$ and $\frac{\partial \Delta S}{\partial t}^{\text{upper}}$, demand for *Soil* is considered steady.

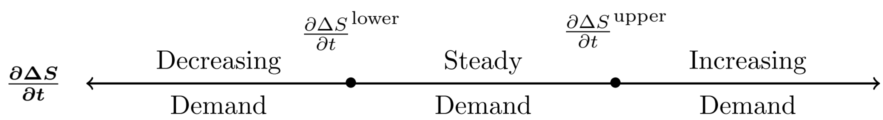


Figure 7: Soil Demand Changes From $\frac{\partial \Delta S}{\partial t}$

However, when Beans are *Sown* in all or almost all *Soil* in consecutive *Seasons* (*i.e.*, $t-1$ and $t-2$), $\frac{\partial \Delta S}{\partial t}$ can inaccurately measure changing demand for *Soil*. In cases where $\frac{\partial \Delta S}{\partial t}$ can inaccurately indicate changing demand for *Soil*, the difference in time it took for the Beans to be *Sown* over the previous two *Seasons* ($\Delta E_t^u$), such that $\Delta E_t^u \in \mathbb{Z}$, provides a more accurate measurement.

In order to measure $\Delta E_t^u$, Beanstalk logs the time of the last *Sow* in each *Season* ($\Delta E_t^{u^{\text{last}}}$), such that $\Delta E_t^{u^{\text{last}}} \in \mathbb{N}$, as the difference between the Ethereum timestamp of the last *Sow* in $t$ ($E_t^{u^{\text{last}}}$) and $E_t$.

We define $\Delta E_t^{u^{\text{last}}}$ for a given $E_t^{u^{\text{last}}}$ and $E_t$ as:

$$\Delta E_t^{u^{\text{last}}} = E_t^{u^{\text{last}}} - E_t$$

If the *Soil Rate* was below the *Minimum Soil Rate* (*i.e.*, $R^S < R^{S^{\text{min}}}$) at the end of both $t-1$ and $t-2$ and $\frac{\partial \Delta S}{\partial t}$ indicates steady demand, at the beginning of $t$ Beanstalk compares $\Delta E_{t-2}^{u^{\text{last}}}$ with the difference in time between the first *Sow* in $t-1$ where the rate of change of $R^S$ from *Season* to *Season* ($\frac{\partial R^S}{\partial t}$), such that $\frac{\partial R^S}{\partial t} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, indicates steady demand ($E_{t-1}^{u^{\text{first}}}$) and $E_{t-1}$.

Beanstalk calculates $\frac{\partial R^S}{\partial t}$ from the percent change in $R^S$ over both $t-1$ and $t-2$ ($\Delta R_{t-1}^S$ and $\Delta R_{t-2}^S$), such that $\Delta R_{t-1}^S,\ \Delta R_{t-2}^S \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$.

We define $\Delta R_t^S$ for a given $R^S$ at the start and end of $t$ ($R_t^{S^{\text{start}}}$ and $R_t^{S^{\text{end}}}$), such that $R_t^{S^{\text{start}}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+,\ j \leq 10^6\}$ and $R_t^{S^{\text{end}}} \in \{j \times 10^{-6} \mid j \in \mathbb{N},\ j \leq 10^6\}$, as:

$$\Delta R_t^S = \frac{R_t^{S^{\text{end}}}}{R_t^{S^{\text{start}}}}$$

We define $\frac{\partial R^S}{\partial t}$ for a given $\Delta R^S_{t-1}$ and $\Delta R^S_{t-2}$ as:

$$\frac{\partial R^S}{\partial t} = \frac{\Delta R^S_{t-2}}{\Delta R^S_{t-1}}$$

Beanstalk requires a $\frac{\partial R^S}{\partial t}$ level to be set: $\frac{\partial R^S}{\partial t}^{\text{upper}}$, such that $\frac{\partial R^S}{\partial t}^{\text{upper}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, below which Beanstalk logs $E^{u^{\text{first}}}_{t-1}$.

We define $\Delta E^u_t$ for a given $\Delta E^{u^{\text{last}}}_{t-2}$, $E^{u^{\text{first}}}_{t-1}$ and $E_{t-1}$ as:

$$\Delta E^u_t = \Delta E^{u^{\text{last}}}_{t-2} - (E^{u^{\text{first}}}_{t-1} - E_{t-1})$$

If the above condition is met, changing demand for *Soil* is measured by $\Delta E^u_t$. Beanstalk requires two $\Delta E^u_t$ levels to be set: (1) $\Delta E^{u^{\text{lower}}}_t$, below which demand for *Soil* is considered decreasing, and (2) $\Delta E^{u^{\text{upper}}}_t$, above which demand for *Soil* is considered increasing, such that $\Delta E^{u^{\text{lower}}}_t$, $\Delta E^{u^{\text{upper}}}_t \in \mathbb{Z}$. When $\Delta E^u_t$ is between $\Delta E^{u^{\text{lower}}}_t$ and $\Delta E^{u^{\text{upper}}}_t$, demand for *Soil* is considered steady.

Thus, Beanstalk measures changing demand for *Soil*.

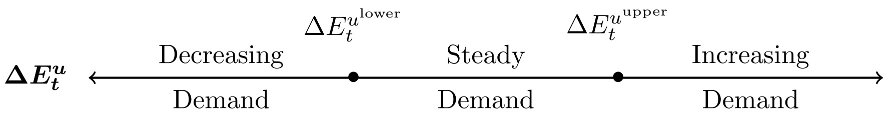


Figure 8: Soil Demand Changes From $\Delta E^u_t$

#### 8.4.6 Current State

The *Weather* change is also determined in part by the current state of Beanstalk with respect to ideal equilibrium.

We define the current state of Beanstalk as the combination of its direction and acceleration with respect to ideal equilibrium. With two potential directions and three potential accelerations, Beanstalk has six potential current states:

- Accelerating away from ideal equilibrium;
- Steady away from ideal equilibrium;
- Decelerating away from ideal equilibrium;
- Accelerating toward ideal equilibrium;
- Steady toward ideal equilibrium; and
- Decelerating toward ideal equilibrium.

| | | **Acceleration** | | |
|---|---|---|---|---|
| | **Current State** | **Decelerating** | **Steady** | **Accelerating** |
| **Direction** | **Away From** | Decelerating Away From | Steady Away From | Accelerating Away From |
| | **Toward** | Decelerating Toward | Steady Toward | Accelerating Toward |

Figure 9: Current State

#### 8.4.7 Optimal State

Beanstalk changes the *Weather* at the beginning of $t$ in an attempt to move Beanstalk from its current state with respect to ideal equilibrium into an optimal state. When the current state is an optimal state, Beanstalk does not change the *Weather*.

An optimal state of Beanstalk is an optimal current state as determined by its current debt level.

We define an optimal state of Beanstalk as accelerating toward ideal equilibrium or either steady or decelerating toward ideal equilibrium. When $R^D$ is excessively high or low, an optimal state is accelerating toward ideal equilibrium. When $R^D$ is reasonably high or low, an optimal state is either steady or decelerating toward ideal equilibrium.

| Optimal State | $\boldsymbol{R^D}$: Excessively Low Debt | Reasonably Low Debt | Reasonably High Debt | Excessively High Debt |
|---|---|---|---|---|
| | Accelerating Toward | Steady or Decelerating Toward | Steady or Decelerating Toward | Accelerating Toward |

Figure 10: Optimal State

#### 8.4.8 Weather Changes

The *Weather* change at the beginning of $t$ is determined by $R^D$ and the current state of Beanstalk with respect to ideal equilibrium.

When $R^D > R^{D^{\text{upper}}}$ (*i.e.*, debt is excessively high):

- If the current state is accelerating or steady away from ideal equilibrium, the *Weather* is raised 3%;
- If the current state is decelerating away from or toward ideal equilibrium, the *Weather* is raised 1%;
- If the current state is steady toward ideal equilibrium, the *Weather* is kept constant; or
- If the current state is accelerating toward ideal equilibrium, the *Weather* is lowered 1%.

When $R^{D^*} < R^D < R^{D^{\text{upper}}}$ (*i.e.*, debt is reasonably high):

- If the current state is accelerating or steady away from ideal equilibrium, the *Weather* is raised 3%;
- If the current state is decelerating away from ideal equilibrium, the *Weather* is raised 1%;
- If the current state is decelerating or steady toward ideal equilibrium, the *Weather* is kept constant; or
- If the current state is accelerating toward ideal equilibrium, the *Weather* is lowered 1%.

When $R^{D^{\text{lower}}} < R^D < R^{D^*}$ (*i.e.*, debt is reasonably low):

- If the current state is accelerating or steady away from ideal equilibrium, the *Weather* is lowered 3%;
- If the current state is decelerating away from ideal equilibrium, the *Weather* is lowered 1%;
- If the current state is decelerating toward ideal equilibrium, the *Weather* is kept constant;
- If the current state is steady toward ideal equilibrium, the *Weather* is raised 1%; or
- If the current state is accelerating toward ideal equilibrium, the *Weather* is raised 3%.

When $R^D < R^{D^{\text{lower}}}$ (*i.e.*, debt is excessively low):

- If the current state is accelerating or steady away from ideal equilibrium, the *Weather* is lowered 3%;
- If the current state is decelerating away from ideal equilibrium, the *Weather* is lowered 1%;
- If the current state is decelerating toward ideal equilibrium, the *Weather* is kept constant;
- If the current state is steady toward ideal equilibrium, the *Weather* is raised 1%; or
- If the current state is accelerating toward ideal equilibrium, the *Weather* is raised 3%.

Thus, Beanstalk changes the *Weather* to regularly cross the price of Ø1 over its value peg during long run decreases and increases in demand for Beans.

| | | $\boldsymbol{R^D}$ | | | |
|---|---|---|---|---|---|
| | **Weather Changes** | **Excessively Low Debt** | **Reasonably Low Debt** | **Reasonably High Debt** | **Excessively High Debt** |
| **Current State** | **Accelerating Away From** | -3 | -3 | 3 | 3 |
| | **Steady Away From** | -3 | -3 | 3 | 3 |
| | **Decelerating Away From** | -1 | -1 | 1 | 1 |
| | **Decelerating Toward** | 0 | 0 | 0 | 1 |
| | **Steady Toward** | 1 | 1 | 0 | 0 |
| | **Accelerating Toward** | 3 | 3 | -1 | -1 |

Figure 11: Weather Changes From Current State and $R^D$

| | | | $\boldsymbol{R^D}$ | | | |
|---|---|---|---|---|---|---|
| | **Weather Changes** | | **Excessively Low Debt** | **Reasonably Low Debt** | **Reasonably High Debt** | **Excessively High Debt** |
| $\boldsymbol{\bar{P}}$ **& Demand Changes** | $\boldsymbol{\bar{P}_{t-1} > 1}$ | **Increasing** | -3 | -3 | -1 | -1 |
| | | **Steady** | -3 | -3 | 0 | 0 |
| | | **Decreasing** | -1 | -1 | 0 | 1 |
| | $\boldsymbol{\bar{P}_{t-1} < 1}$ | **Increasing** | 0 | 0 | 1 | 1 |
| | | **Steady** | 1 | 1 | 3 | 3 |
| | | **Decreasing** | 3 | 3 | 3 | 3 |

Figure 12: Weather Changes From $\bar{P}$, Demand Changes and $R^D$

## 8.5 Sale on Uniswap

Beanstalk sells newly minted Beans on Uniswap during long run increases in demand for Beans when increasing the Bean supply and lowering the *Weather* has not crossed the price of $\emptyset$1 over its value peg.

If at the beginning of $t$, $\bar{P}_{t-1} > 1$ and $R^D < R^{D^*}$, it is *Raining*. If it *Rains* for 24 consecutive *Seasons*, the 24th *Season* – and each successive *Season* in which it continues to *Rain* – is a *Season of Plenty*. At the beginning of a *Season of Plenty*, Beanstalk crosses the price of $\emptyset$1 over its value peg by minting additional Beans and selling them directly on Uniswap. Proceeds of $Y$ from the sale are distributed to *Stalk* owners at the beginning of $t$ in proportion to their *Stalk* ownership when it started *Raining*. Also at the beginning of a *Season of Plenty*, all *Pods* that grew from *Sown* $\emptyset$ before it started *Raining* ripen and become *Harvestable*.

The number of Beans that are minted and sold on Uniswap to cross the price of $\emptyset$1 over the value peg ($\Delta b_{t-1}$) is calculated from the difference between the optimal number of Beans in the $\emptyset$:$Y$ liquidity pool at the end of the previous *Season* ($b^*_{t-1}$), such that $\Delta b_{t-1},\ b^*_{t-1} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, and $b_{t-1}$.

We define $b^*_{t-1}$ for a given $b_{t-1}$, $y_{t-1}$, number of $X$ in the $X$:$Y$ liquidity pool at the end of the previous *Season* ($x_{t-1}$), such that $x_{t-1} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, and number of $Y$ in the $X$:$Y$ liquidity pool at the end of the previous *Season* ($n_{t-1}$), such that $n_{t-1} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, as:

$$b^*_{t-1} = \sqrt{\frac{b_{t-1} \times y_{t-1} \times x_{t-1}}{n_{t-1}}}$$

We define $\Delta b_{t-1}$ for a given $b^*_{t-1}$ and $b_{t-1}$ as:

$$\Delta b_{t-1} = \frac{\lceil b^*_{t-1} - b_{t-1} \rceil}{0.9985}$$

In a *Season of Plenty*, $m_t$ for a given number of *Unharvestable Pods* that grew prior to the start of the *Rain* ($D_r$), such that $D_r \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, $a_t$, $\Delta\bar{b}_{t-1}$, and $\Delta b_{t-1}$ is:

$$m_t = D_r + a_t + \Delta\bar{b}_{t-1} + \Delta b_{t-1}$$

Thus, Beanstalk regularly crosses the price of $\emptyset$1 over its value peg.

# 9 Economics

Beanstalk is designed from economic first principles to increase censorship resistance, liquidity and stability over time.

## 9.1 Concentration

A design that lowers the Gini coefficient[17] of Beans and *Stalk* over time is essential to censorship resistance.

Beanstalk does not require a pre-mine. The first 100 Beans are created when the `sunrise()` function is called to begin the 1st *Season* as part of the Beanstalk deployment.

*Deposited* $\emptyset$ and $\Lambda$ have their *Stalk* diluted relative to total outstanding *Stalk* every *Season*. Therefore, newly minted Beans are more widely distributed over time.

[17] wikipedia.org/wiki/Gini_coefficient

## 9.2 Credit

Beanstalk is credit based and only fails if it can no longer attract creditors. A reasonable level of debt, a strong credit history and a competitive interest rate attract creditors.

Beanstalk changes the *Weather* to return $R^D$ to $R^{D^*}$ while regularly crossing the price of $\not{0}$1 over its value peg. Beanstalk acts more aggressively when $R^D$ is excessively high or low.

Beanstalk never defaults on debt and is willing to issue *Pods* every *Season*.

## 9.3 Marginal Rate of Substitution

There are a wide variety of opportunities Beanstalk has to compete with for creditors. Beanstalk does not define an optimal *Weather*, but instead adjusts it to move closer to ideal equilibrium.

## 9.4 Friction

Minimizing the cost of using Beans and barriers to working on the *Bean Farm* maximize utility for users and appeal to creditors.

A Beanstalk website code base is published alongside the Beanstalk code base to ensure widespread availability of user-friendly cost-efficient access to Beanstalk.

The FIFO *Pod Harvest* schedule allows smaller *Bean Farmers* to participate in peg maintenance and decreases the benefit of large scale price manipulation. The combination of non-expiry, the FIFO *Harvest* schedule and transferability enables *Bean Farmers* to *Sow* Beans as efficiently as possible. By maximizing the efficiency of the *Soil* market, Beanstalk minimizes the cost to service debt, the durations and magnitudes of price deviations below its value peg and excess Bean minting.

## 9.5 Equilibrium

Equilibrium is a state of equivalent marginal quantity supplied and demanded. Beanstalk affects the marginal supply of and demand for Beans to regularly cross the equilibrium price of $\not{0}$1 over its value peg.

While Beanstalk can arbitrarily increase the Bean supply when the equilibrium price of $\not{0}$1 is above its value peg, Beanstalk cannot arbitrarily decrease the Bean supply when the equilibrium price of $\not{0}$1 is below it. Beanstalk relies on the codependence between the equilibria of Beans and *Soil* to work around this limitation.

In order to *Sow* Beans, they must be acquired (*i.e.*, marginal demand for *Soil* affects marginal demand for Beans). The marginal demand for *Soil* is a function of the *Weather* and the Bean price. By changing the *Weather*, Beanstalk affects decreases in the Bean supply.

## 9.6 Incentives

*Stalk* owners and *Bean Farmers* are financially motivated actors. Beanstalk-native financial incentives consistently increase censorship resistance, liquidity and stability over time.

The *Stalk System* incentivizes (1) leaving assets *Deposited* in the *Silo* continuously by creating opportunity cost to *Withdrawing* assets from the *Silo*, and (2) adding value to the ₿:$Y$ liquidity pool by rewarding more *Seeds* to *Deposited* $\Lambda$ than *Deposited* ₿.

Beanstalk is governed by *Stalk* owners. Anyone with *Stalk* stands to profit from future growth of Beanstalk, but are not owed anything by Beanstalk. The inability for the submitter of and voters for a $BIP$ to *Withdraw* from the *Silo* during its *Voting Period* further aligns their interests with Beanstalk.

When $\bar{P}_t$ is below its value peg, there is an incentive to *Withdraw* assets from the *Silo* and the ₿:$Y$ Uniswap liquidity pool. The combination of the *Stalk System* and 24 *Season Freeze* on *Withdrawals* reduces this incentive significantly.

When $\bar{P}_t$ is above its value peg, there is an incentive to buy Beans to earn a portion of the upcoming Bean inflation. This is exacerbated when $R^D$ is lower. The combination of the commitment to automatically cross the price of ₿1 over its value peg and pay $Y$ proceeds from the sale to *Stalk* owners at the start of the *Rain* who still own *Stalk*, and the 24 *Season Freeze* on *Withdrawals*, removes this incentive entirely during *Seasons* where $R^D$ is excessively low, and reduces it significantly otherwise.

Thus, Beanstalk consistently increases censorship resistance, liquidity and stability over time.

# 10 Risk

There are numerous risks associated with Beanstalk. This is not an exhaustive list.

The Beanstalk code base is unaudited. The Beanstalk peg maintenance mechanism is novel. Neither have been tested in the "real world" prior to the initial Beanstalk deployment. The open source nature of Beanstalk means that others can take advantage of any bugs, flaws or deficiencies in Beanstalk and launch identical or very similar stablecoin implementations.

A decentralized implementation of Beanstalk has three external dependencies: (1) a trustless computer network that supports fungible token standards, (2) a decentralized exchange protocol that runs on the trustless computer network, and (3) a first generation stablecoin native to the trustless computer network that offers convertibility to the value peg and trades on the decentralized exchange protocol.

To date, the Ethereum blockchain is the most developed decentralized smart contract platform and has an active community, the ERC-20 Standard is the most widely used fungible token standard, Uniswap is the largest Ethereum-native decentralized exchange protocol by volume,[18] and USDC is the largest convertible USD stablecoin by market capitalization and volume on Uniswap.[19] In general, open source protocols with large amounts of value on them (*e.g.*, Ethereum, Uniswap and USDC) are high value targets for exploits. Long track records indicate security. We assume the security of the Ethereum blockchain, ERC-20 Standard and Uniswap.

There is no guarantee the centralized operators of USDC will not ban USDC from Uniswap, although this would cause significant financial self-harm. The operators of USDC may alter their convertibility policy, which would negatively affect the accuracy of USDC as a price source for USD.

The Beanstalk price oracle contains exposure to custody risk for the underlying collateral of $X$ (*e.g.*, USDC). However, in theory, in cases where the collateral is lost or feared to be lost, the price of $X$ will most likely fall below $V$. This would cause some short run excess inflation of the Bean supply until $X$ is replaced in the price oracle, but would not otherwise directly affect Beanstalk.

[18] defiprime.com/dex-volume
[19] info.uniswap.org/

# 11 Future Work

Beanstalk is a work in progress. There are a number of potential improvements that can be incorporated into Beanstalk as one or more $BIPs$.

*Stalk* and *Seeds* will become liquid shortly after the initial Beanstalk deployment. Deposits also can become liquid assets to further decrease friction.

$\Lambda$ *Deposits* can be rewarded *Seeds* and *Stalk* for Uniswap trading fees.

The mechanism to measure changing demand for *Soil*, in cases where $\frac{\partial \Delta S}{\partial t}$ can inaccurately indicate changing demand for *Soil*, can be further refined by lowering the trigger below $R^{S^{\min}}$.

Additional $X$ and Ethereum-native decentralized exchanges can be incorporated into $\bar{P}^{X:Y}$.

As Uniswap v3[20] demonstrates security and attracts liquidity, Beanstalk can incorporate an existing $X$:$Y$ v3 liquidity pool into $\bar{P}^{X:Y}$ and a new ₿:$Y$ v3 liquidity pool into $\bar{P}^{₿:Y}$ and the *Silo*.

The Beanstalk website can be improved to include more live data and analytical tools.

In the future, we expect unique Beanstalk with different value pegs and on different decentralized networks.

# 12 Parameters for Initial Beanstalk

The following are the parameters of the initial Beanstalk at the time of deployment:

- $w_1 = 1$;
- $R^{S^{\min}} = 0.1\%$;
- $R^{S^{\max}} = 25\%$;
- $R^{D^{\text{lower}}} = 5\%$;
- $R^{D^*} = 15\%$;
- $R^{D^{\text{upper}}} = 25\%$;
- $\frac{\partial \Delta S}{\partial t}^{\text{lower}} = 95\%$;
- $\frac{\partial \Delta S}{\partial t}^{\text{upper}} = 105\%$;
- $\frac{\partial R^S}{\partial t}^{\text{upper}} = 105\%$;
- $\Delta E_t^{u^{\text{lower}}}$ = -60; and
- $\Delta E_t^{u^{\text{upper}}} = 60$.

[20] uniswap.org/blog/uniswap-v3/

# 13 Glossary

The following conventions are used throughout this paper:

- Lower case letters are unique values;
- Upper case letters are totals or rates;
- Subscripts are time; and
- Superscripts are modifiers.

The following variables are used throughout this paper:

$\emptyset$ - Beans, the Beanstalk ERC-20 Standard stablecoin.

$\emptyset$:$Y$ - A new Uniswap liquidity pool of $\emptyset$ and $Y$.

\$ - US Dollars.

$a^{BIP}$ - The award for submitting a $BIP$ that gets accepted.

$A^{BIP}$ - The total $a^{BIP}$ for all passed $BIP$.

$a^q$ - The award for successfully committing an approved $BIP$.

$A^q$ - The total $a^q$ for all committed $BIP$.

$a_t$ - The award for successfully calling the `sunrise()` function for $t$.

$B$ - The total Bean supply.

*Bean Farm* - The *Silo* and *Field*.

*Bean Farmers* - Beanstalk creditors.

$BIP$ - A *Beanstalk Improvement Proposal*.

$b_h$ - The number of Beans in the $\emptyset$:$Y$ liquidity pool at the time of $h$.

$B_t$ - The Bean supply at the start of $t$.

$b_{t-1}$ - The number of $\emptyset$ in the $\emptyset$:$Y$ liquidity pool at the end of the previous *Season*.

$\bar{b}_{t-1}$ - The time weighted average number of Beans in the $\emptyset$:$Y$ liquidity pool over the previous *Season*.

$b^*_{t-1}$ - The optimal number of Beans in the $\emptyset$:$Y$ liquidity pool at the end of the previous *Season*.

$\bar{b}^*_{t-1}$ - The time weighted average optimal number of Beans in the $\emptyset$:$Y$ liquidity pool over the previous *Season*.

*Convert* - Contribute additional $Y$ to the $\emptyset$:$Y$ liquidity pool and exchange *Deposited* $\emptyset$ for *Deposited* $\Lambda$.

Current State - The combination of Beanstalk's direction and acceleration with respect to ideal equilibrium.

$C_t$ - A *Silo Member's* total *Seeds*.

$c^{\emptyset}_t$ - The *Seeds* for an updated $\lambda^{\emptyset}$.

$c^{\Lambda}_t$ - The *Seeds* for an updated $\lambda^{\Lambda}$.

$d$ - The number of *Pods* that grow from *Sown* Ø.

$D$ - The total number of unripened *Pods*.

DAO - A Decentralized Autonomous Organization.

DeFi - Decentralized Finance.

*Deposit* - Assets in the *Silo*.

*Depositors* - Wallets that *Deposit* assets in the *Silo*.

$D_r$ - The number of *Unharvestable Pods* that grew prior to the start of the *Rain*.

$\Delta b_{t-1}$ - The number of Beans that are minted and sold on Uniswap to cross the price of Ø1 over the value peg at the beginning of a *Season of Plenty*.

$\Delta \bar{b}_{t-1}$ - The time weighted average shortage or excess of Beans in the Ø:$Y$ liquidity pool over the previous *Season*.

$\Delta E^u_t$ - The difference in time it took for the Beans to be *Sown* over the previous two *Seasons*.

$\Delta E^{u^{\text{last}}}_t$ - The time of the last *Sow* in $t$.

$\Delta E^{u^{\text{lower}}}_t$ - The $\Delta E^u_t$ level below which demand for *Soil* is considered decreasing.

$\Delta E^{u^{\text{upper}}}_t$ - The $\Delta E^u_t$ level above which demand for *Soil* is considered increasing.

$\Delta S$ - The change in *Soil* from the beginning to the end of each *Season*.

$\Delta R^S_{t-1}$ - The percent change in $R^S$ over the previous *Season*.

$\frac{\partial \Delta S}{\partial t}$ - The rate of change of $\Delta S$ from *Season* to *Season*.

$\frac{\partial \Delta S}{\partial t}^{\text{lower}}$ - The $\frac{\partial \Delta S}{\partial t}$ level below which demand for *Soil* is considered decreasing.

$\frac{\partial \Delta S}{\partial t}^{\text{upper}}$ - The $\frac{\partial \Delta S}{\partial t}$ level above which demand for *Soil* is considered increasing.

$\frac{\partial R^S}{\partial t}$ - The rate of change of $R^S$ from *Season* to *Season*.

$\frac{\partial R^S}{\partial t}^{\text{upper}}$ - The $\frac{\partial R^S}{\partial t}$ level below which Beanstalk logs $E^{u^{\text{first}}}_{t-1}$.

$E$ - Ethereum block timestamps.

ETH - Ether.

$E_1$ - The timestamp in the Ethereum block containing the Beanstalk deployment.

$E_{BIP}$ - The end of a *Voting Period*.

$E_f$ - The timestamp of last *Unpause*.

$E_q$ - The timestamp a $BIP$ is committed.

$E_t$ - The timestamp in the Ethereum block containing the accepted `sunrise()` function call for $t$.

$E^{\text{min}}_t$ - The minimum timestamp Beanstalk accepts a `sunrise()` function call for $t$.

$E^{u^{\text{first}}}_{t-1}$ - The first *Sow* in $t-1$ such that $\frac{\partial R^S}{\partial t}$ would still be considered steady.

*Field* - The Beanstalk lending facility.

FIFO - First in, first out.

*Frozen* - Not able to move.

$G$ - The total *Burnt Beans* over all *Seasons*.

*Harvest* - Redeem ripened *Pods*.

*Harvestable Pods* - Redeemable *Pods*.

$h$ - A $\Lambda$ *Deposit*.

$i$ - The *Season* of *Deposit*.

Ideal Equilibrium - A state where the Bean price and Beanstalk debt level are both stable at their optimal levels.

$K_t$ - A *Silo Member's* total *Stalk* during $t$.

$k_t^{\emptyset}$ - The *Stalk* during $t$ for an updated $\lambda^{\emptyset}$.

$k_t^{\Lambda}$ - The *Stalk* during $t$ for an updated $\lambda^{\Lambda}$.

LP tokens - Uniswap liquidity pool tokens.

$l_i^{\Lambda}$ - A *Silo Member's* list of $\Lambda$ *Deposits* during $i$.

$\lambda^{\emptyset}$ - A map of a *Silo Member's* $\emptyset$ *Deposited* each *Season*.

$\lambda^{\Lambda}$ - A map of the amounts and Bean-denominated amounts of a *Silo Member's* $\Lambda$ *Deposited* each *Season*.

$\Lambda$ - LP tokens for the $\emptyset$:$Y$ Uniswap liquidity pool.

$\Lambda_h$ - The total $\Lambda$ at the time of each $h$.

$M$ - The total Beans minted over all *Seasons*.

Optimal State - The optimal current state of Beanstalk.

$m_t$ - The Beans minted at the beginning of $t$.

$n_{t-1}$ - The number of $Y$ in the $X$:$Y$ liquidity pool at the end of the previous *Season*.

*Pause* - Temporarily prevent the `sunrise()` function call from being accepted.

*Pod* - The Beanstalk debt asset, redeemable for $\emptyset 1$ when ripened.

*Pod Rate* - The Beanstalk debt level relative to the Bean supply.

$\bar{P}$ - The Beanstalk oracle price of $\emptyset 1$.

$\bar{P}^{\emptyset:Y}$ - The TWAP of the $\emptyset$:$Y$ liquidity pool.

$\bar{P}^{X:Y}$ - The TWAP of the $X$:$Y$ liquidity pool.

$\bar{P}_t$ - The TWAP of $\emptyset 1$ over the current *Season*.

$\bar{P}_{t-1}^{\emptyset:Y}$ - The TWAP of the $\emptyset$:$Y$ liquidity pool over the previous *Season*.

$\bar{P}_{t-1}^{X:Y}$ - The TWAP of the $X$:$Y$ liquidity pool over the previous *Season*.

*Rain* - It is *Raining* if $\bar{P}_{t-1} > 1$ and $R^D < R^{D^*}$.

$R^D$ - The *Pod Rate*.

$R^{D^{\text{lower}}}$ - The $R^D$ level below which debt is considered excessively low.

$R^{D^{\text{upper}}}$ - The $R^D$ level above which debt is considered excessively high.

$R^{D^*}$ - The optimal $R^D$ level.

$R^S$ - The *Soil Rate*.

$R^{S^{\max}}$ - The *Maximum Soil Rate*.

$R^{S^{\min}}$ - The *Minimum Soil Rate*.

$R_t^{S^{\text{end}}}$ - The $R^S$ at the end of $t$.

$R_t^{S^{\text{start}}}$ - The $R^S$ at the start of $t$.

$S$ - The *Soil* supply.

*Season* - A unit of time.

*Seeds* - An ERC-20 Standard token that grows one *Stalk* each *Season*.

*Silo* - The Beanstalk DAO.

*Silo Member* - A *Stalk* owner.

*Stalk* - The ERC-20 Standard Beanstalk governance token.

*Stalk System* - A set of *Stalk* policies.

*Soil* - The current number of Beans that can be *Sown* in exchange for *Pods*.

*Soil Rate* - The portion of the total Bean supply Beanstalk is currently willing to remove in exchange for debt.

*Sow* - Lend.

$s_t$ - The *Soil* minted at the beginning of $t$.

$S_t^{\max}$ - The *Maximum Soil* at the beginning of $t$.

$S_t^{\min}$ - The *Minimum Soil* at the beginning of $t$.

$S_t^{\text{start}}$ - The *Soil* supply at the beginning of $t$.

$S_{t-1}^{\text{end}}$ - The *Soil* supply at the end of the previous *Season*.

TWAP - Time weighted average price.

$t$ - A *Season*.

$t^{'}$ - The *Season* of the last *Unpause*.

$u$ - *Sown* ₿.

$U$ - The total *Sown* ₿ over all *Seasons*.

*Unharvestable* - Not redeemable.

*Unpause* - Resume the acceptance of `sunrise()` function calls.

USD - US Dollars.

USDC - US Dollar Coins.

$V$ - The value peg.

*Voting Period* - The time interval a $BIP$ is considered.

$w$ - The percentage of additional Beans ultimately *Harvested* from 1 *Sown* ₿.

$X$ - An existing ERC-20 Standard first generation stablecoin that pegs to $V$.

$X$:$Y$ - An existing Uniswap liquidity pool of $X$ and $Y$.

$x_{t-1}$ - The number of $X$ in the $X$:$Y$ liquidity pool at the end of the previous *Season*.

$Y$ - A decentralized ERC-20 Standard token.

$y_{t-1}$ - The number of $Y$ in the $\emptyset$:$Y$ liquidity pool at the end of the previous *Season*.

$z_i^{\emptyset}$ - The $\emptyset$ a *Silo Member Deposits* during $i$.

$z_i^{\Lambda}$ - The amount of $\Lambda$ a *Silo Member Deposits* during $i$.

$z_i^{\Lambda:\emptyset}$ - The Bean-denominated amount of $\Lambda$ a *Silo Member Deposits* during $i$.