Beanstalk: A Permissionless Fiat Stablecoin Protocol

Publius
beanstalk.publius@protonmail.com
bean.money

Published: August 6, 2021
Modified: December 8, 2022
Whitepaper Version: 2.3.0
Code Version: 2.3.0[1]

"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[2]

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 sufficiently low volatility in value relative to an arbitrary value peg (*e.g.*, 1 US Dollar (USD, $), 100 Satoshis and 1 oz of Gold). To date, flawed stablecoin implementations sacrifice the main benefits of trustless computing by requiring a custodian or limit their potential supply and utility by imposing collateral requirements, and suffer from noncompetitive carrying costs. A stablecoin that does not compromise on decentralization nor require collateral, has competitive carrying costs, and trends toward more stability and liquidity, will unlock the potential of DeFi. We propose an Ethereum[3]-native, fiat stablecoin protocol that issues an ERC-20 Standard[4] token that fulfills these requirements. A decentralized autonomous organization (DAO) governed by a variable supply, yield generating token simultaneously provides security, dampens price volatility and encourages consistent liquidity growth. Beanstalk uses a decentralized credit facility, network-native price oracle, variable supply and self-adjusting interest rate, to regularly cross the stablecoin price over its value peg without requiring action from users.

1 github.com/BeanstalkFarms/Beanstalk
2 founders.archives.gov/documents/Hamilton/01-02-02-1167
3 ethereum.org
4 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[5] and Ethereum) present the next economic and technological frontiers: trustless goods and services. Instead of requiring users to trust (1) a rent-seeking third party to write secure code, run it on secure computer servers and perform fair system administration, or (2) concentrated risk-taking counterparties, 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. Diverse sets of users and network maintenance participants remove concentrated counterparty risk, which creates decentralization. The combination of permissionlessness, sound economics and decentralization creates censorship resistance, which is fundamental to trustlessness. Potential applications built on top of well designed trustless networks are infinite.

A key promise of trustless computer networks is the widespread use of financial goods and services without the need for trust-providing, rent-seeking central authorities, or concentrated counterparties. However, as blockchains that support trustless networks are adopted, the values of their native assets (*e.g.*, Bitcoin and Ether (ETH)) change radically. To date, the practicality of using DeFi technologies for real economic activity is limited by the lack of a trustless network-native asset with competitive carrying costs, low-volatility endogenous value and deep liquidity.

A stablecoin protocol generates a fungible network-native asset and attempts to keep its price volatility sufficiently low relative to an arbitrary value peg. Stablecoin utility is a function of trustlessness, carrying costs, stability and liquidity. A stablecoin's trustlessness, carrying costs, stability and liquidity are primarily functions of the source of its value. Current implementations fail to deliver a stablecoin that is (1) sufficiently decentralized and permissionless, (2) unrestricted by collateral requirements and their associated noncompetitive carrying costs, (3) sufficiently low in price volatility relative to its value peg and (4) highly liquid, due to a lack of endogenous value creation.

1.1 Convertible Stablecoins

Non-network-native exogenous value convertible stablecoin protocols (*e.g.*, US Dollar Coin[6] (USDC), Tether[7] (USDT), Wrapped Bitcoin,[8] and RenBTC[9]) issue stablecoins they claim are collateralized by, and require a custodian that facilitates the convertibility to, non-network-native exogenous value worth at least 100% of total outstanding protocol liabilities. Stablecoin protocols that offer convertibility to non-network-native assets function as low-volatility permissioned bridges between their respective networks and the rest of the world. Arbitrage opportunities created by convertibility ensure the price of the network-native asset is rarely above or below the value of the custodied value when accounting for frictions around conversions. However, users of non-network-native exogenous value convertible stablecoins sacrifice permissionlessness and carry entirely: third parties custody the non-network-native assets, can freeze the network-native assets unilaterally and can retain yield earned on collateral. The absence of protocol-native opportunities for carry limits liquidity.

[5] bitcoin.org
[6] circle.com/usdc
[7] tether.to
[8] wbtc.network
[9] renproject.io

Network-native exogenous value convertible stablecoin protocols (*e.g.*, Maker[10] (DAI) and Abracadabra[11]) use excess network-native collateral to remove most points of centralization. Overcollateralization removes most risk associated with the volatility of the collateral but by necessity requires the introduction of rent payments in order to prevent the value of the stablecoin from trending towards the value of the underlying collateral. The combination of collateral requirements and rent payments significantly limits the potential supply of these stablecoins. Liquity[12] is an ideal simple iteration of a network-native exogenous value convertible stablecoin protocol, without any points of centralization and with protocol-native positive carry. In order to remove rent payments, Liquity does not target an exact price for its stablecoin, LUSD. The potential supply of LUSD is limited by the value of trustless network-native value.

Despite the shortcomings of exogenous value convertible stablecoin implementations, demand for their USD implementations continues to increase rapidly. Over the twelve months prior to the initial deployment of Beanstalk, the total market capitalization of exogenous value convertible USD stablecoins increased more than 500% to over $100 Billion.[13] However, despite this rapid increase in supply, the borrowing rates on exogenous value convertible USD stablecoins have historically been higher[14] than borrowing rates on USD.[15] Noncompetitive carrying costs are due to collateral requirements. Businesses built on trustless primitives cannot compete with businesses built on centralized systems due to noncompetitive carrying costs on low-volatility network-native trustless assets.

To date, implementations of purely endogenous value convertible stablecoins (*e.g.*, Terra[16]) have failed. While hybrid value convertible stablecoins (*e.g.*, FRAX[17]) have demonstrated some efficacy at peg maintenance at high proportions of exogenous value, their supply is limited by network-native exogenous value.

1.2 Non-convertible Stablecoins

Non-convertible stablecoin protocols adjust themselves mechanically to return the price of their stablecoin to their value peg without convertibility to collateral. It is impossible to keep a stablecoin price equal to its value peg without low-friction convertibility. Non-convertible stablecoin protocols without collateral requirements have the potential to create endogenous value that facilitates trustlessness, competitive carrying costs and deep liquidity at the expense of volatility.

Rebasing stablecoin protocols (*e.g.*, Ampleforth[18]) have shown efficacy at crossing their stablecoin prices over their value pegs, but without the regularity, low volatility or liquidity necessary to create utility. Extreme negative carrying costs during decreases in demand exacerbate difficulty of use.

The value of fiat currency is derived from the credit of its issuer and its utility. Utility of fiat currency is a function of trustlessness, carrying costs, stability and liquidity. Decentralized credit can be used to issue a permissionless fiat stablecoin with competitive carrying costs, low volatility and deep liquidity.

To date, however, implementations of fiat stablecoin protocols have failed to regularly cross their stablecoin prices over their value pegs with sufficiently low volatility due to poorly designed peg maintenance mechanisms or seigniorage models that disproportionately reward speculators at the expense of stablecoin utility.

10 makerdao.com
11 abracadabra.money
12 liquity.org
13 stablecoinindex.com/marketcap
14 app.aave.com/markets
15 newyorkfed.org/markets/reference-rates/effr
16 allcryptowhitepapers.com/terra-whitepaper
17 frax.finance
18 ampleforth.org

1.3 Beanstalk

Beanstalk uses a dynamic peg maintenance mechanism to regularly cross the price of 1 Bean (₿) – the Beanstalk ERC-20 Standard fiat stablecoin – over its value peg without centralization or collateral requirements. 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 decreased volatility over time. Regularly crossing the price of ₿1 over its value peg creates the opportunity to regularly buy and sell Beans at their value peg.

Beanstalk consists of five interconnected components: (1) a decentralized timekeeping and execution facility, (2) a decentralized governance facility, (3) a decentralized credit facility, (4) a decentralized exchange (DEX), and (5) an interface to interact with other Ethereum-native protocols via Beanstalk. Beanstalk-native financial incentives are used to coordinate the components to (1) create a stablecoin with competitive carrying costs, (2) regularly cross the price of ₿1 over its value peg during both long run decreases and increases in demand for Beans, and (3) attract deep liquidity, in a cost-efficient, permissionless and decentralized fashion.

Beanstalk is designed from economic first principles to create a useful trustless fiat currency. Over time, trustlessness, stability and liquidity increase, while carrying costs decrease but remain competitive. 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 composability and fungible token standards (*e.g.*, Ethereum) with a network-native automated market maker (AMM) decentralized exchange (*e.g.*, Uniswap[19] and Curve[20]) is required to implement a decentralized stablecoin.

Stablecoin protocols that offer convertibility to the non-network-native asset they are pegged to reliably bridge the value of non-network-native assets to the network. Beanstalk leverages the existence of non-network-native exogenous value convertible stablecoins that trade on AMMs to create a new permissionless stablecoin with a non-network-native value peg.

Beanstalk was inspired by Empty Set Dollar.[21] The failures of Empty Set Dollar and similar stablecoin implementations provided invaluable information that influenced the design of Beanstalk.

[19] uniswap.org
[20] curve.fi
[21] emptyset.finance

3 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 to create utility for end users. Low barriers to and variety in work enable a diverse set of participants. A diverse set of well incentivized workers can create 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 can join the *Farm* to use Beanstalk and profit from participation in protocol maintenance. Governance of Beanstalk upgrades, Bean peg maintenance and use of Beans take place on the *Farm*.

The *Farm* has five primary components: the *Sun*, *Silo*, *Field*, *Market* and *Depot*. Beanstalk-native financial incentives coordinate the components to create a stalwart system of governance, regularly cross the price of $\not\Beta 1$ over its value peg, consistently grow Bean liquidity and maximize composability, without collateral.

The *Sun* offers payment for participation in timekeeping and execution. Time on the *Farm* is kept in *Seasons*. Anyone can earn Beans for successfully calling the `sunrise` function to begin the next *Season* at the top of each hour.

The *Silo* is the Beanstalk DAO. The *Silo* offers passive yield opportunities to owners of $\not\Beta$ and other assets (λ) on the *Deposit Whitelist* (Λ) (*i.e.*, $\not\Beta \subset \lambda \in \Lambda$) for participation in governance of Beanstalk upgrades and passive contribution to security, stability and liquidity. Anyone can become a *Stalkholder* by *Depositing* λ into the *Silo* to earn *Stalk*. *Stalkholders* govern Beanstalk upgrades and are rewarded with Beans when the Bean supply increases. Active contributions to peg maintenance within the *Silo* earn additional *Stalk*.

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

The *Market* offers 0-fee trading to anyone using the Ethereum network.

The *Depot* facilitates interactions with other Ethereum-native protocols through Beanstalk in a single transaction.

4 Sun

The Beanstalk governance and peg maintenance mechanisms require a protocol-native timekeeping mechanism and regular code execution on the Ethereum blockchain. The *Sun* creates a cost-efficient protocol-native timekeeping mechanism and incentivizes 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. The first *Season* begins when a successful transaction on the Ethereum blockchain that includes a `sunrise` function call is mined. When Beanstalk accepts the `sunrise` function call, the necessary code is executed.

Beanstalk only accepts one `sunrise` function call per *Season*. Beanstalk accepts the first `sunrise` function call provided 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 ($E_t^{\min}$), $\forall$ $E_t^{\min}$ such that $1 < t$, and E_1 is:

$$E_t^{\min} = 3600\left(\left\lfloor \frac{E_1}{3600} \right\rfloor + t\right)$$

The cost to execute the `sunrise` function 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 25 Beans and compounds 1% every additional second that elapses past $E_t^{\min}$ for 300 seconds.

The award for successfully calling the `sunrise` function for t (a_t), such that $a_t \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$ and $1 < t$, in a block with a given timestamp (E_t) is:

$$a_t = 25 \times 1.01^{\min\{E_t - E_t^{\min},\ 300\}}$$

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

5 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, stability and liquidity. *Stalkholders* earn passive yield from participation in governance of Beanstalk upgrades and passive contributions to security, stability and liquidity. Active contributions to peg maintenance within the *Silo* earn additional *Stalk*.

5.1 The Stalk System

The *Stalk System* decentralizes ownership over time and creates Beanstalk-native financial incentives to (1) align DAO voters' interests with the health of Beanstalk, (2) leave assets *Deposited* in the *Silo*, and (3) allocate liquidity in ways that benefit Beanstalk.

Anyone can become a *Stalkholder* by *Depositing* assets on the *Deposit Whitelist* into the *Silo* to earn *Stalk* and *Seeds*. *Stalk* and *Seeds* are not liquid. Every *Season*, 1×10^{-4} additional *Stalk* *Grows* from each *Seed*. *Grown Stalk* become *Stalk* when *Mown*. *Grown Stalk* are automatically *Mown* each time a *Stalkholder* interacts with the Silo (*i.e.*, *Deposit*, *Withdraw*, *Convert*, *Transfer*, *Plant* and *Enroot*), or when they call the `mow` function.

Stalkholders 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 *Stalkholder* are proportional to their *Stalk* holdings relative to total outstanding *Stalk*. *Stalk* holdings become less concentrated over time.

5.2 Deposit Whitelist

Any ERC-20 Standard token can be added to and removed from Λ via Beanstalk governance. $\mathbb{B}$ is always on the *Deposit Whitelist*.

In order for a given λ to be added to Λ Beanstalk requires (1) its token address, (2) a function to calculate the flash-loan-resistant Bean-denominated-value (BDV) for a given number of λ *Deposited*, $(f^{\lambda}(z^{\lambda}))$, such that $z^{\lambda} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$, f^{λ}: $\{j \times 10^{-\lambda} \mid j \in \mathbb{Z}^{+}\} \to \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$, where z^{λ} is the number of λ *Deposited*, (3) the number of *Stalk* per BDV of λ *Deposited* (k^{λ}), such that $k^{\lambda} \in \{j \times 10^{-4} \mid j \in \mathbb{Z}^{+}\}$, and (4) the number of *Seeds* per BDV of λ *Deposited* (c^{λ}), such that $c^{\lambda} \in \mathbb{Z}^{+}$, to be stored.

5.3 Deposits, Withdrawals, Transfers and Conversions

λ can be *Deposited* into, *Withdrawn* from and *Converted* within, the *Silo* at any time. Upon *Withdrawal*, assets are *Frozen* until the end of the current *Season* and for ξ, such that $\xi \in \mathbb{N}$, additional *Seasons*.[22]

Beanstalk rewards *Stalk* and *Seeds* to *Depositors* immediately upon *Depositing* λ into the *Silo* based on its BDV when *Deposited*, k^{λ} and c^{λ}.

The number of *Stalk*, *Seeds*, and *Stalk* from *Seeds* rewarded for a *Deposited* asset must be forfeited upon its *Withdrawal* from the *Silo*, or included in its *Transfer* to another address.

Conversions of *Deposited* λ to *Deposited* λ' are permissioned by a *Convert Whitelist*. *Conversions* can be added or removed from the *Convert Whitelist* via Beanstalk governance. In order for a given *Convert* to be added to the *Convert Whitelist*, Beanstalk requires (1) the from token address, (2) the to token address, (3) a list of conditions under which the *Conversion* is and is not permitted, and (4) a function to determine the number of λ' received for *Converting* a given number of λ $(f^{\lambda \to \lambda'}(z^{\lambda}))$, such that $f^{\lambda \to \lambda'}$: $\{j \times 10^{-\lambda} \mid j \in \mathbb{Z}^{+}\} \to \{j \times 10^{-\lambda'} \mid j \in \mathbb{Z}^{+}\}$, where z^{λ} is the number of λ *Converted*.

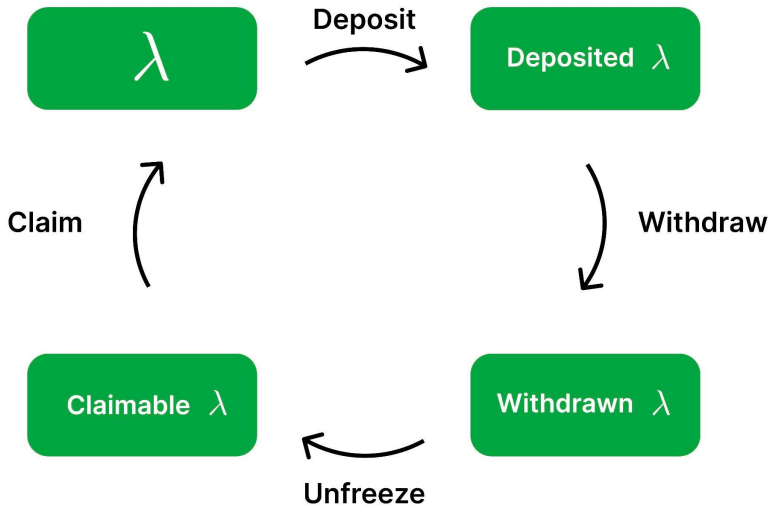

Figure 1: Silo

[22] bean.money/bip-9

5.4 Calculating Stalk and Seeds

A *Stalkholder's* total *Stalk* is the sum of the *Stalk* for each of their *Deposits* and *Earned* $\not{B}$ ($\eta^{\not{B}}$), such that $\eta^{\not{B}} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$. *Earned* $\not{B}$ are Beans paid to a *Stalkholder* after the last *Season* the *Stalkholder* called the `plant` function (η).[23,24,25] *Earned* $\not{B}$ automatically earn *Stalk*. The next time the *Stalkholder* calls the `plant` function, *Earned* $\not{B}$ are *Deposited* in the current *Season* and the associated *Seeds* are *Planted* to start *Growing Stalk*.

When a *Stalkholder Deposits* λ, they update the total number of λ *Deposited* during *Season* i (Z_i^λ) and its total BDV when *Deposited* (L_i^λ), such that $Z_i^\lambda,\ L_i^\lambda \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, as $Z_i^\lambda \mathrel{+}= z^\lambda$ and $L_i^\lambda \mathrel{+}= f^\lambda(z^\lambda)$. Beanstalk stores a map of each *Stalkholder's Deposits* that are still in the *Silo*, from *Stalkholder* to token address to *Season* to *Deposit* totals (*i.e.*, $(Z_i^\lambda,\ L_i^\lambda)$).

The *Stalk* for a given *Deposit* are determined by its duration of *Deposit*, BDV when *Deposited*, k^λ, c^λ and the last *Season* the *Stalkholder Mowed* their *Grown Stalk* ($\varkappa$).

The *Stalk* during t for a given *Deposit* of a *Stalkholder* that last *Mowed* their *Grown Stalk* in $\varkappa$ (k_t^λ), such that $k_t^\lambda \in \{j \times 10^{-10} \mid j \in \mathbb{Z}^+\}$, is:

$$k_t^\lambda = L_i^\lambda \left(k^\lambda + \frac{c^\lambda(\varkappa - i)}{10000} \right)$$

A *Stalkholder's* total *Stalk* during t (K_t), such that $K_t \in \{j \times 10^{-10} \mid j \in \mathbb{N}\}$, is:

$$K_t = \sum_{\lambda \in \Lambda} \sum_{i=1}^{\varkappa} k_t^\lambda + \eta^{\not{B}}$$

The *Grown Stalk* from *Seeds* that can be *Mown* during t to start earning Bean seigniorage for a given *Deposit* of a *Stalkholder* that last *Mowed* their *Grown Stalk* in $\varkappa$ (g_t^λ), such that $g_t^\lambda \in \{j \times 10^{-10} \mid j \in \mathbb{N}\}$, is:

$$g_t^\lambda = L_i^\lambda \left(\frac{c^\lambda(t - \varkappa)}{10000} \right)$$

A *Stalkholder's* total *Grown Stalk* that can be *Mown* during t (G_t), such that $G_t \in \{j \times 10^{-10} \mid j \in \mathbb{N}\}$, is:

$$G_t = \sum_{\lambda \in \Lambda} \sum_{i=1}^{\varkappa} g_t^\lambda$$

The *Seeds* for a given *Deposit* are determined by its BDV when *Deposited* and c^λ.

The *Seeds* during t for a given *Deposit* (c_t^λ), such that $c_t^\lambda \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, is:

$$c_t^\lambda = c^\lambda L_i^\lambda$$

A *Stalkholder's* total *Seeds* during t (C_t), such that $C_t \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, is:

$$C_t = \sum_{\lambda \in \Lambda} \sum_{i=1} c_t^\lambda$$

The *Plantable Seeds* associated with a *Stalkholder's* $\eta^{\not{B}}$ that can be *Planted* to start earning *Grown Stalk* (η^c), such that $\eta^c \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, is:

$$\eta^c = c^{\not{B}} \times \eta^{\not{B}}$$

[23] bean.money/bip-0
[24] bean.money/bpp-0
[25] bean.money/bip-21

When a *Stalkholder Withdraws* λ, they must forfeit the number of *Stalk*, *Seeds*, and *Stalk* from *Seeds* rewarded to the assets being *Withdrawn* and update their map accordingly.

When a *Stalkholder Transfers* λ, they must include the number of *Stalk*, *Seeds*, and *Stalk* from *Seeds* rewarded to the assets being *Transferred* and update their maps accordingly.

When a *Stalkholder Converts* a *Deposit*, they update its *Season* of *Deposit* to retain its *Grown Stalk* from *Seeds*, and BDV if it is higher. A maximum of $0.0c^{\lambda}\%$ of *Stalk* for a *Converted Deposit* is forfeited due to rounding. When *Converting* a λ *Deposit Deposited* before $\left\lceil t\left(1-\frac{c^{\lambda'}}{c^{\lambda}}\right)\right\rceil$ to a $\emptyset$ *Deposit*, some *Stalk* from *Seeds* may be forfeited due to rounding. When *Converting* multiple λ *Deposits*, their *Seasons* of *Deposit* are averaged together, weighted by their *BDVs*, and rounded up.

5.5 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 *Beanstalk Improvement Proposal* (*BIP*), join the *Silo* and cast their votes, with the ability to be quickly upgraded in cases of emergency.

5.5.1 Participation

Any λ owner can become a *Stalkholder* and participate in Beanstalk governance by *Depositing* λ into the *Silo* to earn *Stalk*.

Any *Stalkholder* that owns more than $K^{\min}$, such that $K^{\min} \in \{j \times 10^{-10} \mid j \in \mathbb{N},\ j \leq 10^{6}\}$, of total outstanding *Stalk* can submit a *BIP* via the *Beanstalk Community Multisig*[26] (*BCM*). In the future, as the ownership concentration of *Stalk* decreases, we expect a *BIP* to lower this threshold.

Beanstalk only accepts votes in favor of *BIPs*. A *Stalkholder's* vote is counted in proportion to their *Stalk* when the *BIP* is submitted to Snapshot.[27]

The award for submitting a *BIP* that gets accepted (a^{BIP}), such that $a^{\text{BIP}} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, is determined by the submitter 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* except for a more reasonable a^{BIP}.

5.5.2 Voting Period

A *Voting Period* opens when a *BIP* is submitted to Snapshot and ends at approximately the beginning of the 169th *Season* after it is submitted to Snapshot, 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* at the time the *BIP* was submitted to Snapshot that still exists votes in favor of the *BIP*, it fails; and
- More than half of the total outstanding *Stalk* at the time the *BIP* was submitted to Snapshot that still exists votes in favor of the *BIP*, it passes.

[26] bean.money/bcm-process
[27] snapshot.org/#/beanstalkdao.eth

If at any time before the end of the *Voting Period* more than two-thirds of the total outstanding *Stalk* at the time the *BIP* was submitted to Snapshot that still exists votes in favor of the *BIP*, it can be committed to the Ethereum blockchain.

5.5.3 Pause

In case of a particularly dangerous vulnerability to Beanstalk, the *Silo* can *Pause* or *Unpause* Beanstalk via *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_Ψ) during *Season* $t^{'}$, we define $E_t^{\min} \ \forall \ E_t^{\min}$ such that $t^{'} < t$ as:

$$E_t^{\min} = 3600\left(\left\lfloor \frac{E_\Psi}{3600} \right\rfloor + t - t^{'}\right)$$

5.5.4 Beanstalk Improvement Proposals

Beanstalk implements EIP-2535.[28] Beanstalk is a diamond with multiple facets. Beanstalk supports multiple simultaneous *BIPs* with independent *Voting Periods*.

A BIP has four inputs: (1) a ternary for whether Beanstalk should *Pause* or *Unpause*, where a 1 *Pauses* Beanstalk, a 2 *Unpauses Beanstalk*, and a null input does neither, (2) a list of facets and functions to add and remove upon commit, (3) a function to run upon commit, and (4) the Ethereum address of the contract with (3).

5.5.5 Beanstalk Community Multisig

The *BCM* address has the exclusive and unilateral ability to *Pause* and *Unpause* Beanstalk, and submit and commit *BIPs*. The *BCM* is a 5-of-9 Safe[29] multisig wallet with anonymous signers consisting of community members and contributors to Beanstalk. The *BCM* will provide sufficient notice of the submission, its contents and the submission time before submitting a *BIP* to Snapshot. In the future, we expect *BIPs* will reimplement permissionless governance and revoke these abilities from the *BCM*.

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

6 Field

The Beanstalk peg maintenance mechanism requires the ability to borrow Beans. The *Field* is the Beanstalk credit facility.

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

[28] eips.ethereum.org/EIPS/eip-2535
[29] app.safe.global/eth:0xa9bA2C40b263843C04d344727b954A545c81D043

6.1 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*. $\not\Diamond 1$ is *Sown* in one *Soil*. Beanstalk permanently removes *Sown* $\not\Diamond$ from the Bean supply.

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

6.2 Pods

Pods are the primary debt asset of Beanstalk. Beanstalk never defaults on debt: *Pods* automatically *Yield* from *Sown* $\not\Diamond$ and never expire.

In the future, when the average price of $\not\Diamond 1$ is above its value peg over a *Season*, *Pods Ripen* and become *Harvestable* (redeemable) for $\not\Diamond 1$ at anytime. *Pods Ripen* on a first in, first out (FIFO) basis: *Pods Yielded* from Beans that are *Sown* first *Ripen* into *Harvestable Pods* first. *Pod* holders can *Harvest* their *Harvestable Pods* anytime by calling the `harvest` function. There is no penalty for waiting to *Harvest Pods*.

Pods are transferable. In practice, *Pods* are non-callable zero-coupon bonds with priority for maturity represented as a place in line. The number of *Pods* that *Yield* from *Sown* $\not\Diamond$ is determined by the *Temperature*.

6.3 Temperature

We define the *Temperature* (h), such that $h \in \mathbb{Z}^+$, as the percentage of additional Beans ultimately *Harvested* from 1 *Sown* $\not\Diamond$.

The number of *Pods* (d) that *Yield* from a given number of *Sown* $\not\Diamond$ (u), such that $d,\ u \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, *Sown* with a given h is:

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

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

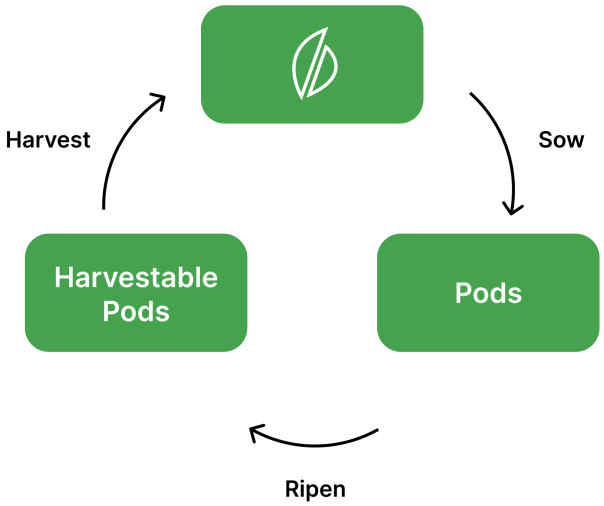

Figure 2: Field

7 Barn

The *Barn* is the Beanstalk recapitalization facility, being used for the Beanstalk *Replant*.[30,31]

Anytime there is *Available Fertilizer* (defined below) in the *Barn*, any owner of ETH or USDC can buy *Fertilizer* from Beanstalk. The *Humidity* is the interest rate on *Fertilizer* purchases.

7.1 Fertilizer

Fertilizer is a limited debt issuance. *Fertilizer* automatically *Fertilizes Sprouts* and never expires.

We define *Available Fertilizer* ($\mathfrak{V}$) as the number of *Fertilizer* that can be bought from Beanstalk in exchange for 1 USDC each, *Active Fertilizer* ($\mathfrak{A}$) as the number of *Fertilizer* that have been bought but have not *Fertilized* all associated *Sprouts*, and *Used Fertilizer* ($\mathfrak{U}$), such that $\mathfrak{V}$, $\mathfrak{A}$, $\mathfrak{U} \in \mathbb{N}$, as the number of *Fertilizer* that have been bought and *Fertilized* all associated *Sprouts*.

In the future, when the average price of $\mathbb{1}$ is above its value peg over a *Season*, *Active Fertilizer* *Fertilizes Sprouts* such that they become *Rinsable* (redeemable) for $\mathbb{1}$ at anytime. *Active Fertilizer* *Fertilizes* a pro-rata portion of *Sprouts*, by *Fertilizer*. *Fertilizer* owners can *Rinse* their *Rinsable* *Sprouts* anytime by calling the `rinse` function. There is no penalty for waiting to *Rinse Sprouts*.

Fertilizer is transferable. In practice, *Fertilizer* is a non-callable zero-coupon pari passu bond without a fixed maturity. The number of *Sprouts* that *Fertilizer* ultimately *Fertilizes* is dependent on the *Humidity* at its time of purchase.

[30] bean.money/bfp-72
[31] bean.money/barn

7.2 Humidity

We define the *Humidity* ($\mathfrak{H}$), such that $\mathfrak{H} \in \{j \times 10^{-1} \mid j \in \mathbb{Z}^+\}$, as 1 less than the number of Beans ultimately *Fertilized* from 1 *Fertilizer* divided by 100.

The number of *Sprouts* ($\mathfrak{d}$), such that $\mathfrak{d} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, ultimately *Fertilized* by *Available Fertilizer* purchased with given $\mathfrak{H}$ ($\mathfrak{V}_{\mathfrak{H}}$), such that $\mathfrak{V}_{\mathfrak{H}} \in \mathbb{Z}^+$, is:

$$\mathfrak{d} = \mathfrak{V}_{\mathfrak{H}} \times \left(1 + \frac{\mathfrak{H}}{100}\right)$$

The *Humidity* is constant each *Season*. The *Humidity* is 500 prior to the *Replant*, after which it is 250 for a full *Season* and then decreases by 0.5 each *Season* until it reaches 20.

7.3 Recapitalization

Beanstalk uses the proceeds from the sale of *Fertilizer* to recapitalize value stolen from *Stalkholders* in the April 17th, 2022 governance exploit (the *Exploit*). Beanstalk will sell enough *Fertilizer* to fully recapitalize all non-Beanstalk-native value stolen from *Stalkholders*.

The proportion of a *Stalkholder's Stalk* and *Seeds* at the end of the block prior to the *Exploit* that have been *Revitalized* and can be *Enrooted* to begin earning Bean seigniorage and *Grown Stalk*, respectively, is a function of the percentage of *Fertilizer* sold.

Non-Beanstalk-native and Beanstalk-native value stolen from *Stalkholders* are recapitalized simultaneously via *Unripe* assets. *Unripe* assets entitle holders to an associated number of *Ripe* assets (*i.e.*, $\emptyset$ and BEAN:3CRV Curve LP tokens (Φ), such that $\Phi \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$). The number of *Ripe* assets associated with a given *Unripe* asset increases as more *Fertilizer* is sold. Holders of *Unripe* assets can *Chop* them and receive a portion of the associated *Ripe* asset at anytime. The portion of *Ripe* assets that can be received by *Chopping* a given *Unripe* asset increases as the percentage of *Sprouts Fertilized* increases. Claims to future *Ripe* assets are forfeited upon *Chopping* the *Unripe* asset.

7.3.1 Available Fertilizer

The number of *Available Fertilizer* is the difference between the total *Fertilizer* ($\mathfrak{F}$) and total *Fertilizer* sold ($\mathfrak{S}$), such that $\mathfrak{F},\ \mathfrak{S} \in \mathbb{Z}^+$. $\mathfrak{F}$ is a function of the current total *Unripe* Φ ($\mathfrak{Z}^{\Phi}$) and the total *Unripe* Φ at the time of *Replant* ($\mathfrak{Z}^{\Phi}_{\otimes}$), such that $\mathfrak{Z}^{\Phi},\ \mathfrak{Z}^{\Phi}_{\otimes} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$. $\mathfrak{S}$ is the sum of *Active Fertilizer* and *Used Fertilizer*.

We define $\mathfrak{F}$ for a given $\mathfrak{Z}^{\Phi}$ and $\mathfrak{Z}^{\Phi}_{\otimes}$ as:

$$\mathfrak{F} = \frac{7.7 \times 10^7 \times \mathfrak{Z}^{\Phi}}{\mathfrak{Z}^{\Phi}_{\otimes}}$$

We define $\mathfrak{S}$ for a given $\mathfrak{A}$ and $\mathfrak{U}$ as:

$$\mathfrak{S} = \mathfrak{A} + \mathfrak{U}$$

Therefore, we define $\mathfrak{V}$ for a given $\mathfrak{F}$ and $\mathfrak{S}$ as:

$$\mathfrak{V} = \mathfrak{F} - \mathfrak{S}$$

7.3.2 Revitalized Stalk and Seeds

Upon *Replant*, *Stalkholders* at the end of the block prior to the *Exploit* received a portion of their *Stalk*, *Seeds* and *Plantable Seeds* at the end of the block prior to the *Exploit* based on the percentage of *Fertilizer* sold prior to *Replant* ($\mathfrak{X}_{\otimes}$), such that $\mathfrak{X}_{\otimes} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$. As the percentage of *Fertilizer* sold ($\mathfrak{X}$), such that $\mathfrak{X} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, increases, additional *Stalk* and *Seeds* are *Revitalized* and can be *Enrooted*. *Revitalized Stalk* and *Revitalized Seeds* become *Stalk* and *Seeds* respectively, upon being *Enrooted*.

We define $\mathfrak{X}$ for a given $\mathfrak{S}$ and $\mathfrak{F}$ as:

$$\mathfrak{X} = \frac{\mathfrak{S}}{\mathfrak{F}}$$

A *Stalkholder's Stalk* upon *Replant* ($K_{\otimes}$) given $\mathfrak{X}_{\otimes}$ and their *Stalk* at the end of the block prior to the *Exploit* ($K_{\odot}$), such that $K_{\otimes},\ K_{\odot} \in \{j \times 10^{-10} \mid j \in \mathbb{Z}^+\}$, is:

$$K_{\otimes} = \mathfrak{X}_{\otimes} \times K_{\odot}$$

A *Stalkholder's Seeds* upon *Replant* ($C_{\otimes}$) given $\mathfrak{X}_{\otimes}$, their *Seeds* at the end of the block prior to the *Exploit* ($C_{\odot}$) and their *Plantable Seeds* at the end of the block prior to the *Exploit* ($\eta^c_{\odot}$), such that $C_{\otimes},\ C_{\odot},\ \eta^c_{\odot} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, is:

$$C_{\otimes} = \mathfrak{X}_{\otimes} \times (C_{\odot} + \eta^c_{\odot})$$

The number of *Revitalized Stalk* (φ^K_t), such that $\varphi^K_t \in \{j \times 10^{-10} \mid j \in \mathbb{Z}^+\}$, and *Revitalized Seeds* (φ^C_t), such that $\varphi^C_t \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, that can be *Enrooted* by a *Stalkholder* during t are functions of the change in $\mathfrak{X}$ ($\Delta\mathfrak{X}$), such that $\Delta\mathfrak{X} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, between (1) the *Season* they last called the `enroot` function (φ) or (2) the *Replant* if they have never *Enrooted* their *Revitalized Stalk* and *Revitalized Seeds* (*i.e.*, $\varphi = 0$), and t, and $K_{\odot}$ or $C_{\odot}$ and $\eta^c_{\odot}$, respectively.

We define $\Delta\mathfrak{X}$ for a given *Stalkholder* that last *Enrooted* their *Revitalized Stalk* and *Revitalized Seeds* in φ as:

$$\Delta\mathfrak{X} = \begin{cases} \mathfrak{X}_t - \mathfrak{X}_{\varphi} & \text{if } \varphi > 0 \\ \mathfrak{X}_t - \mathfrak{X}_{\otimes} & \text{else} \end{cases}$$

We define φ^K_t for a given $\Delta\mathfrak{X}$ and $K_{\odot}$ as:

$$\varphi^K_t = \Delta\mathfrak{X} \times K_{\odot}$$

We define φ^C_t for a given $\Delta\mathfrak{X}$ and $C_{\odot}$ as:

$$\varphi^C_t = \Delta\mathfrak{X} \times (C_{\odot} + \eta^c_{\odot})$$

7.3.3 Unripe Assets

Holders of Beans at the end of the block prior to the *Exploit* received *Unripe* $\not{\Diamond}$ ($\mathfrak{z}^{\not{\Diamond}}$), such that $\mathfrak{z}^{\not{\Diamond}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, at a 1:1 ratio. [32] Holders of λ not *Deposited* at the end of the block prior to the *Exploit* received *Unripe* Φ ($\mathfrak{z}^{\Phi}$), such that $\mathfrak{z}^{\Phi} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, at a ratio of 1 $\mathfrak{z}^{\Phi}$ per BDV of λ held at the end of the block prior to the *Exploit*. Holders of λ *Deposited* at the end of the block prior to the *Exploit* received $\mathfrak{z}^{\Phi}$ at a ratio of 1 $\mathfrak{z}^{\Phi}$ per the maximum of the BDV of λ *Deposits* at the end of the block prior to the *Exploit* and at the time of *Deposit*, per *Deposit*.

[32] snapshot.org/#/beanstalkdao.eth/proposal/0xe47741c4bfa4ac97ad23bbec0db8b9a5f2efc3e1737b309476d90611698193f4

7.3.4 Ripe Assets

The number of *Ripe* assets (*i.e.*, *Ripe* $\mathbb{\phi}$ ($\mathfrak{R}^{\mathbb{\phi}}$), such that $\mathfrak{R}^{\mathbb{\phi}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$, and *Ripe* Φ ($\mathfrak{R}^{\Phi}$), such that $\mathfrak{R}^{\Phi} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^{+}\}$), increases as more *Fertilizer* is sold.

The change in *Ripe* $\mathbb{\phi}$ ($\Delta\mathfrak{R}^{\mathbb{\phi}}$) for a given purchase of *Fertilizer* ($\Delta\mathfrak{S}_{\mathfrak{d}}$) is a function of the total *Unripe* $\mathbb{\phi}$ ($\mathfrak{Z}^{\mathbb{\phi}}$), the *Ripe* $\mathbb{\phi}$ prior to the purchase ($\mathfrak{R}^{\mathbb{\phi}}_{<\mathfrak{d}}$), such that $\Delta\mathfrak{R}^{\mathbb{\phi}},\ \mathfrak{Z}^{\mathbb{\phi}},\ \mathfrak{R}^{\mathbb{\phi}}_{<\mathfrak{d}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$, $\mathfrak{F}$, and the *Fertilizer* sold prior to the purchase ($\mathfrak{S}_{<\mathfrak{d}}$), such that $\Delta\mathfrak{S}_{\mathfrak{d}},\ \mathfrak{S}_{<\mathfrak{d}} \in \mathbb{Z}^{+}$.

We define $\Delta\mathfrak{R}^{\mathbb{\phi}}$ for a given $\Delta\mathfrak{S}_{\mathfrak{d}}$, $\mathfrak{Z}^{\mathbb{\phi}}$, $\mathfrak{R}^{\mathbb{\phi}}_{<\mathfrak{d}}$, $\mathfrak{F}$, $\mathfrak{S}_{<\mathfrak{d}}$ as:

$$\Delta\mathfrak{R}^{\mathbb{\phi}} = \frac{\Delta\mathfrak{S}_{\mathfrak{d}} \times (\mathfrak{Z}^{\mathbb{\phi}} - \mathfrak{R}^{\mathbb{\phi}}_{<\mathfrak{d}})}{\mathfrak{F} - \mathfrak{S}_{<\mathfrak{d}}}$$

The change in *Ripe* Φ ($\Delta\mathfrak{R}^{\Phi}$), such that $\Delta\mathfrak{R}^{\Phi} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^{+}\}$, is the result of calling the `calc_token_amount` function on the Curve Zap contract[33] for a given $\Delta\mathfrak{S}_{\mathfrak{d}}$.

We define $\Delta\mathfrak{R}^{\Phi}$ for a given $\Delta\mathfrak{S}_{\mathfrak{d}}$ as:

$$\Delta\mathfrak{R}^{\Phi} = \texttt{calc\_token\_amount}(\Phi,\ [0.866616 \times \Delta\mathfrak{S}_{\mathfrak{d}},\ 0,\ \Delta\mathfrak{S}_{\mathfrak{d}},\ 0],\ \textit{true})$$

7.3.5 Chopping

The percentage of *Ripe* assets received for *Chopping* a pro-rata portion of *Unripe* assets ($\mathfrak{M}$) is a function of the total *Sprouts Fertilized* by *Fertilizer* ($\Delta\mathfrak{D}$) and the total *Unfertilized Sprouts* (*i.e.*, *Sprouts* not yet *Fertilized* by *Active Fertilizer*) ($\mathfrak{D}$), such that $\mathfrak{M},\ \Delta\mathfrak{D},\ \mathfrak{D} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$.

We define $\mathfrak{M}$ for a given $\Delta\mathfrak{D}$ and $\mathfrak{D}$ as:

$$\mathfrak{M} = \frac{\Delta\mathfrak{D}}{\Delta\mathfrak{D} + \mathfrak{D}}$$

The number of Beans received for *Chopping* a given $\mathfrak{z}^{\mathbb{\phi}}$ ($\mathfrak{P}^{\mathbb{\phi}}$), such that $\mathfrak{P}^{\mathbb{\phi}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$, is a function of $\mathfrak{M}$, $\mathfrak{R}^{\mathbb{\phi}}$ and $\mathfrak{Z}^{\mathbb{\phi}}$.

We define $\mathfrak{P}^{\mathbb{\phi}}$ for a given $\mathfrak{z}^{\mathbb{\phi}}$, $\mathfrak{M}$ and $\mathfrak{R}^{\mathbb{\phi}}$ as:

$$\mathfrak{P}^{\mathbb{\phi}} = \frac{\mathfrak{z}^{\mathbb{\phi}} \times \mathfrak{M} \times \mathfrak{R}^{\mathbb{\phi}}}{\mathfrak{Z}^{\mathbb{\phi}}}$$

The number of Φ received for *Chopping* a given $\mathfrak{z}^{\Phi}$ ($\mathfrak{P}^{\Phi}$), such that $\mathfrak{P}^{\Phi} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^{+}\}$, is a function of $\mathfrak{M}$, $\mathfrak{S}$, $\mathfrak{Z}^{\Phi}_{\otimes}$ and $\mathfrak{Z}^{\Phi}$.

We define $\mathfrak{P}^{\Phi}$ for a given $\mathfrak{z}^{\Phi}$, $\mathfrak{M}$, $\mathfrak{S}$, $\mathfrak{Z}^{\Phi}_{\otimes}$ and $\mathfrak{Z}^{\Phi}$ as:

$$\mathfrak{P}^{\Phi} = \frac{\mathfrak{z}^{\Phi} \times \mathfrak{M} \times \mathfrak{S} \times \mathfrak{Z}^{\Phi}_{\otimes}}{\mathfrak{Z}^{\Phi}}$$

Chopped Unripe $\mathbb{\phi}$ and Φ are burned (*i.e.*, $\mathfrak{Z}^{\mathbb{\phi}} \mathrel{-}= \mathfrak{z}^{\mathbb{\phi}}$, $\mathfrak{Z}^{\Phi} \mathrel{-}= \mathfrak{z}^{\Phi}$). $\mathbb{\phi}$ and Φ received for *Chopping* are distributed from *Ripe* $\mathbb{\phi}$ and Φ, respectively (*i.e.*, $\mathfrak{R}^{\mathbb{\phi}} \mathrel{-}= \mathfrak{P}^{\mathbb{\phi}}$, $\mathfrak{R}^{\Phi} \mathrel{-}= \mathfrak{P}^{\Phi}$).

[33] etherscan.io/address/0xA79828DF1850E8a3A3064576f380D90aECDD3359

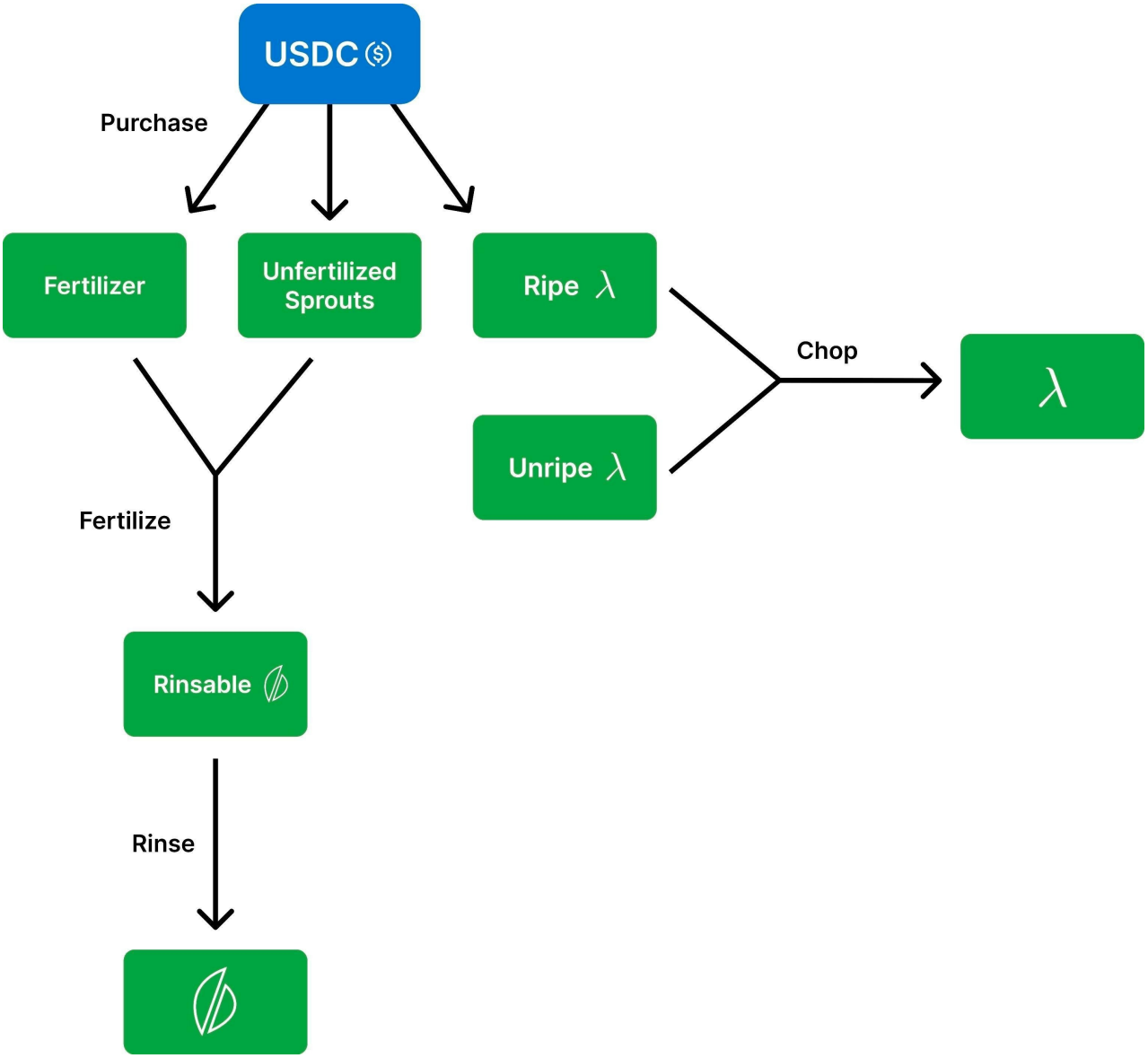

Figure 3: Barn

8 Peg Maintenance

Beanstalk faces the fundamental limitation that it cannot fix the price of $\emptyset 1$ 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 $\emptyset 1$ crosses, and the magnitudes of price deviations from, its value peg. Beanstalk regularly crosses the price of $\emptyset 1$ over its value peg during both long run decreases and increases in demand for Beans.

Beanstalk has four peg maintenance tools available: (1) increase the Bean supply, (2) change the *Soil* supply, (3) change the *Temperature*, and (4) a *Flood* (defined below). 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, and dynamically adjusts the Bean supply, *Soil* supply and *Temperature* 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 $\beta 1$ 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 *Temperature*. Bean supply increases and *Soil* supply changes primarily affect Bean supply. *Temperature* changes 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 both axes are normal and expected. As Beanstalk grows, the durations and magnitudes of deviations decrease.

8.2 Decentralized Price Oracle

One problem native to decentralized stablecoin protocols is the need to be aware of some price without trusting a centralized 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.

When a price source is not native to the network, decentralized price oracles are complicated to build, expensive to maintain and often inaccurate. Beanstalk leverages network-native decentralized AMMs, non-network-native exogenous value and network-native exogenous value convertible stablecoins to remove these complications, costs and inaccuracies almost entirely, and create an immutable, manipulation resistant and decentralized source for the price of a non-Ethereum-native value peg.

Ethereum-native permissionless AMM protocols allow anyone to create new AMMs between at least two ERC-20 Standard tokens. AMMs always offer a price on any size trade, at any time, for a trading fee. AMMs allow continuous trading in either direction by maintaining a liquidity pool of the tokens. The current price is a function of the ratio of the assets in the pool and the AMM pricing formula. Anyone can add liquidity to the pool in exchange for 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 ratio between the sizes of a trade and the liquidity pool. AMMs with larger liquidity pools serve as more robust price sources.

In general, Beanstalk can issue a Bean with a value peg (V) for $\beta 1$ equal to any non-network-native asset (*e.g.*, \$1) with at least one existing ERC-20 Standard convertible stablecoin (x) (*e.g.*, USDC) that (1) offers low-friction convertibility to V, and (2) trades on an AMM against a liquid, decentralized network-native asset with endogenous value (y) (*e.g.*, ETH[34]). To determine the value of $\beta 1$ compared to V, Beanstalk can compare (1) an existing liquidity pool (x:y) (*e.g.*, USDC:ETH) that consists of x and y, and (2) a new liquidity pool (β:y) that consists of Beans and y. The combination of arbitrage opportunities between AMMs and other exchanges, and between x and V, ensures the x:y AMM price closely mirrors the exchange rate between V and y. Beanstalk would consider the price of $\beta 1$ equal to its value peg when the ratios of x:y and β:y are equal.

[34] weth.io

Decentralized systems are never administered by or dependent on a single individual or centralized organization. Beanstalk can leverage an arbitrary x while minimizing exposure to malicious actions from its operators (*e.g.*, censorship) by deriving the price from the ratio between x:y and $\not\Beta$:y.

In practice, Beanstalk never calculates the price of $\not\Beta 1$. Instead, at the beginning of each *Season*, Beanstalk calculates a sum of liquidity and time weighted average shortages or excess of Beans across $\not\Beta$:y liquidity pools on the *Oracle Whitelist* over the previous *Season* ($\Delta B_{\overline{t-1}}$), such that $\Delta B_{\overline{t-1}} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$. Liquidity pools can be added to and removed from the *Oracle Whitelist* via Beanstalk governance.

$\Delta B_{\overline{t-1}}$ can be used to infer the liquidity and time weighted average price of $\not\Beta 1$ compared to V over the previous *Season* ($P_{\overline{t-1}}$), such that $P_{\overline{t-1}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$. If there was a liquidity and time weighted average shortage of Beans across liquidity pools on the *Oracle Whitelist* over the previous *Season* (*i.e.*, $0 < \Delta B_{\overline{t-1}}$), $V < P_{\overline{t-1}}$. If there was a liquidity and time weighted average excess of Beans across liquidity pools on the *Oracle Whitelist* over the previous *Season* (*i.e.*, $\Delta B_{\overline{t-1}} < 0$), $P_{\overline{t-1}} < V$. If there was neither a liquidity and time weighted shortage nor excess of Beans across liquidity pools on the *Oracle Whitelist* over the previous *Season* (*i.e.*, $\Delta B_{\overline{t-1}} = 0$), $P_{\overline{t-1}} = V$.

$\Delta B_{\overline{t-1}} = 0$ for each *Season* that contains a *Pause* and *Unpause*.

Thus, Beanstalk constructs an immutable, manipulation resistant and decentralized price oracle for a non-Ethereum-native value peg.

8.3 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.

Beanstalk does not consider *Burnt* $\not\Beta$, *Sown* $\not\Beta$, *Unfertilized Sprouts* nor *Unharvestable Pods*, but does consider *Rinsable Sprouts* and *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 *BIPs* (A^{BIP}), total awards for all committed *BIPs* (A^q), total Beans minted via *BIP* (B^{BIP}) (*e.g.*, *Fundraisers*), total *Burnt* $\not\Beta$ over all *Seasons* ($N^{\not\Beta}$) and total *Sown* $\not\Beta$ over all *Seasons* (U), such that $A^{\text{BIP}},\ A^q,\ B^{\text{BIP}},\ N^{\not\Beta},\ U \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, as:

$$B = M + A^{\text{BIP}} + A^q + B^{\text{BIP}} - (N^{\not\Beta} + U)$$

We define R^D for a given the total number of *Unharvestable Pods* (D), such that $D \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, 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 or equal to 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 $R^{D^{\text{lower}}} \leq R^D < R^{D^{\text{upper}}}$ and $R^D \neq R^{D^*}$ (*i.e.*, not optimal), R^D is considered reasonable.

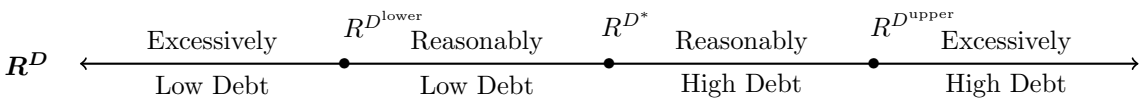

Figure 4: Debt Level

8.4 Position

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

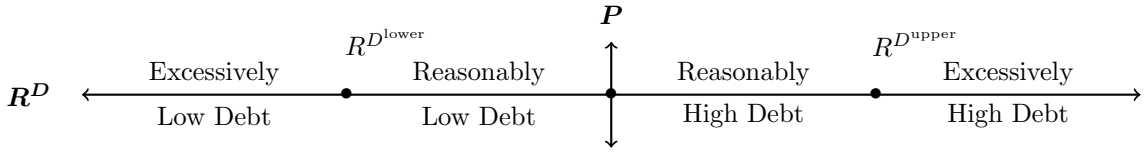

Figure 5: Position

8.5 Direction

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

The direction of change in position of Beanstalk at the beginning of t is considered either toward or away from ideal equilibrium, based on the *Pod Rate* at the end of the previous *Season* (R^D_{t-1}), such that $R^D_{t-1} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, R^D_{t-1} and $P_{\overline{t-1}}$. When $V < P_{\overline{t-1}}$, debt is paid back; when $P_{\overline{t-1}} < V$, debt can only increase or remain constant.

Therefore, when $R^{D^*} < R^D_{t-1}$ (*i.e.*, there was more debt than optimal):

- If $V < P_{\overline{t-1}}$, Beanstalk moves toward ideal equilibrium; and
- If $P_{\overline{t-1}} \leq V$, Beanstalk moves away from ideal equilibrium.

When $R^D_{t-1} \leq R^{D^*}$ (*i.e.*, there was less debt than optimal):

- If $V \leq P_{\overline{t-1}}$, Beanstalk moves away from ideal equilibrium; or
- If $P_{\overline{t-1}} < V$, Beanstalk moves toward ideal equilibrium.

		$\boldsymbol{R^D_{t-1}}$			
	Direction	**Excessively Low Debt**	**Reasonably Low Debt**	**Reasonably High Debt**	**Excessively High Debt**
$\boldsymbol{P_{\overline{t-1}}}$	$\boldsymbol{P_{\overline{t-1}} > 1}$	Away From	Away From	Toward	Toward
	$\boldsymbol{P_{\overline{t-1}} < 1}$	Toward	Toward	Away From	Away From

Figure 6: Direction

8.6 Acceleration

The current state of Beanstalk with respect to ideal equilibrium also is determined by the rate of change of position of Beanstalk at the beginning of each *Season* (*i.e.*, its acceleration).

The acceleration of Beanstalk is considered decelerating, steady or accelerating, based on $P_{\overline{t-1}}$ and changing demand for *Soil*. Demand for *Soil* is considered decreasing, steady or increasing.

When demand for *Soil* is decreasing:

- If $V < P_{\overline{t-1}}$, Beanstalk is decelerating;
- If $P_{\overline{t-1}} < V$, Beanstalk is accelerating;
- If $P_{\overline{t-1}} = V$ and $R^D_{t-1} \leq R^{D^*}$, Beanstalk is accelerating; and
- If $P_{\overline{t-1}} = V$ and $R^{D^*} < R^D_{t-1}$, Beanstalk is decelerating.

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

When demand for *Soil* is increasing:

- If $V \leq P_{\overline{t-1}}$, Beanstalk is accelerating;
- If $P_{\overline{t-1}} < V$, Beanstalk is decelerating;
- If $P_{\overline{t-1}} = V$ and $R^D_{t-1} \leq R^{D^*}$, Beanstalk is decelerating; and
- If $P_{\overline{t-1}} = V$ and $R^{D^*} < R^D_{t-1}$, Beanstalk is accelerating.

		Demand for Soil		
	Acceleration	**Decreasing Demand**	**Steady Demand**	**Increasing Demand**
$P_{\overline{t-1}}$	$\boldsymbol{P_{\overline{t-1}} > 1}$	Decelerating	Steady	Accelerating
	$\boldsymbol{P_{\overline{t-1}} < 1}$	Accelerating	Steady	Decelerating

Figure 7: Acceleration

8.7 Demand for Soil

In order to properly classify its acceleration, Beanstalk must accurately measure changing demand for *Soil*.

The change in *Soil* from the beginning to the end of each *Season* (ΔS_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 ΔS_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 ΔS_t for a given S_t^{start} and *Soil* supply at the end of that *Season* (S_t^{end}), such that $S_t^{\text{end}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, as:

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

We define $\frac{\partial \Delta S}{\partial t}$ for given ΔS_t over the previous two *Seasons*, ΔS_{t-1} and ΔS_{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 or equal to 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 $\frac{\partial \Delta S}{\partial t}^{\text{lower}} \leq \frac{\partial \Delta S}{\partial t} < \frac{\partial \Delta S}{\partial t}^{\text{upper}}$, demand for *Soil* is considered steady.

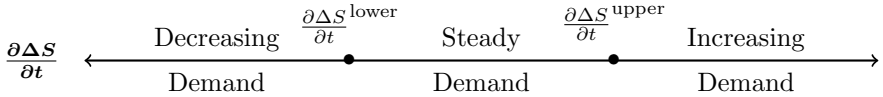

Figure 8: Soil Demand Changes From $\frac{\partial \Delta S}{\partial t}$

However, when Beans are *Sown* in all *Soil* in a *Season* (defined as $S_t^{\text{end}} \leq 1$), $\frac{\partial \Delta S}{\partial t}$ can inaccurately measure changing demand for *Soil*. The first time Beans are *Sown* in all but at most one *Soil* in a *Season*, after one or more *Seasons* where Beans were not *Sown* in all but at most one *Soil*, demand for *Soil* is considered increasing. When Beans are *Sown* in all but at most one *Soil* in consecutive *Seasons* (*i.e.*, $t-1$ and $t-2$), the difference in time it took for the Beans to be *Sown* in all but at most one *Soil* over the previous two *Seasons* (ΔE_t^u), such that $\Delta E_t^u \in \mathbb{Z}$, can provide a more accurate measurement.

In order to measure ΔE_t^u, Beanstalk logs the time of the first *Sow* such that Beans are *Sown* in all but at most one *Soil* in each *Season* ($\Delta E_t^{u^{\text{first}}}$), such that $\Delta E_t^{u^{\text{first}}} \in \mathbb{N}$, as the difference between the Ethereum timestamp of the first *Sow* in t such that there is at most one *Soil* ($E_t^{u^{\text{first}}}$) and E_t.

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

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

If Beans were *Sown* in all but at most one *Soil* in the first 5 minutes of the previous *Season* (*i.e.*, $\Delta E_{t-1}^{u^{\text{first}}} < 300$), demand for *Soil* is considered increasing. If Beans were *Sown* in all but at most one *Soil* in both $t-1$ and $t-2$, but $300 \leq \Delta E_{t-1}^{u^{\text{first}}}$, at the beginning of t Beanstalk compares $\Delta E_{t-1}^{u^{\text{first}}}$ with $\Delta E_{t-2}^{u^{\text{first}}}$ to calculate ΔE_t^u.

We define ΔE_t^u for a given $\Delta E_{t-1}^{u^{\text{first}}}$ and $\Delta E_{t-2}^{u^{\text{first}}}$ as:

$$\Delta E_t^u = \Delta E_{t-2}^{u^{\text{first}}} - \Delta E_{t-1}^{u^{\text{first}}}$$

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

Thus, Beanstalk measures changing demand for *Soil*.

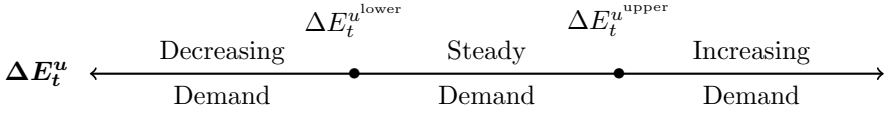

Figure 9: Soil Demand Changes From ΔE_t^u

8.8 Current State

We define the current state of Beanstalk with respect to ideal equilibrium 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 10: Current State

8.9 Optimal State

An optimal state of Beanstalk is an optimal current state 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, the optimal state is accelerating toward ideal equilibrium. When R^D is reasonably high or low, the optimal state is either steady or decelerating toward ideal equilibrium.

	R^D			
Optimal State	**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 11: Optimal State

8.10 Bean Supply

At the beginning of each *Season*, if $V < P_{\overline{t-1}}$, Beanstalk increases the Bean supply based on $\Delta B_{\overline{t-1}}$ in addition to the award for successfully calling the `sunrise` function. Up to two thirds of the additional Bean supply increase is used to pay off debt; the remainder is distributed to *Stalkholders*.

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 $\Delta B_{\overline{t-1}}$ and a_t as:

$$m_t = \max(0,\ \Delta B_{\overline{t-1}}) + a_t$$

The distribution of the additional mint is dependent on $\Delta B_{\overline{t-1}}$, $\mathfrak{D}$ and D. If $0 < \frac{\Delta B_{\overline{t-1}}}{3} \leq \mathfrak{D}$ (*i.e.*, there are at most $\frac{\Delta B_{\overline{t-1}}}{3}$ *Unfertilized Sprouts*), $\frac{\Delta B_{\overline{t-1}}}{3}$ *Sprouts* are *Fertilized* by *Active Fertilizer* and become *Rinsable*. If $0 < \mathfrak{D} < \frac{\Delta B_{\overline{t-1}}}{3}$ (*i.e.*, there are less *Unfertilized Sprouts* than $\frac{\Delta B_{\overline{t-1}}}{3}$), $\mathfrak{D}$ *Sprouts* are *Fertilized* by *Active Fertilizer* and become *Rinsable*.

Therefore, the number of *Unfertilized Sprouts* that are *Fertilized* by *Active Fertilizer* and become *Rinsable* at the beginning of each *Season* ($\Delta\mathfrak{D}_t$), such that $\Delta\mathfrak{D}_t \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, for a given $\Delta B_{\overline{t-1}}$ and $\mathfrak{D}$ is:

$$\Delta\mathfrak{D}_t = \min\left(\max\left(0,\ \frac{\Delta B_{\overline{t-1}}}{3}\right),\ \mathfrak{D}\right)$$

The distribution of the remaining Beans (*i.e.* $\Delta B_{\overline{t-1}} - \Delta\mathfrak{D}_t$) is dependent on D. If $0 < \Delta B_{\overline{t-1}} - \Delta\mathfrak{D}_t < D$ (*i.e.*, there are at most $\Delta B_{\overline{t-1}} - \Delta\mathfrak{D}_t$ *Unharvestable Pods*), $\frac{\Delta B_{\overline{t-1}} - \Delta\mathfrak{D}_t}{2}$ *Pods Ripen* and become *Harvestable* and $\frac{\Delta B_{\overline{t-1}} - \Delta\mathfrak{D}_t}{2}$ newly minted Beans are distributed to *Stalkholders*. If $0 < D < \frac{\Delta B_{\overline{t-1}} - \Delta\mathfrak{D}_t}{2}$ (*i.e.*, there are less *Unharvestable Pods* than $\Delta B_{\overline{t-1}} - \Delta\mathfrak{D}_t$), D *Pods Ripen* and become *Harvestable* and $\Delta B_{\overline{t-1}} - (\Delta\mathfrak{D}_t + D)$ newly minted Beans are distributed to *Stalkholders*.

Therefore, the number of Pods that *Ripen* and become *Harvestable* at the beginning of each *Season* (ΔD_t), such that $\Delta D_t \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, is:

$$\Delta D_t = \min\left(\max\left(0,\ \frac{\Delta B_{\overline{t-1}} - \Delta\mathfrak{D}_t}{2}\right),\ D\right)$$

8.11 Soil Supply

Beanstalk sets the *Soil* supply at the beginning of each *Season*. Beanstalk is willing to issue debt every *Season*. When $V \leq P_{\overline{t-1}}$ the *Soil* supply is based on (1) the number of Pods that *Ripen* and become *Harvestable* at the beginning of the *Season*, (2) the *Temperature* during t (h_t), that $h_t \in \mathbb{Z}^+$, and (3) R^D_{t-1}. When $P_{\overline{t-1}} < V$, the *Soil* supply is also based on $\Delta B_{\overline{t-1}}$.

The minimum *Soil* at the beginning of t ($S^{\min}_t$), such that $S^{\min}_t \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, Beanstalk has outstanding for a given ΔD_t, h_t and R^D_{t-1} is:

$$S^{\min}_t = \begin{cases} \dfrac{0.5 \times \Delta D_t}{1 + \frac{h_t}{100}} & \text{if } R^{D^{\text{upper}}} \leq R^D_{t-1} \\ \dfrac{\Delta D_t}{1 + \frac{h_t}{100}} & \text{if } R^{D^{\text{lower}}} < R^D_{t-1} \\ \dfrac{1.5 \times \Delta D_t}{1 + \frac{h_t}{100}} & \text{else} \end{cases}$$

The *Soil* supply at the beginning of each *Season* (S^{start}_t), such that $S^{\text{start}}_t \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, for a given $\Delta B_{\overline{t-1}}$ and $S^{\min}_t$ is:

$$S^{\text{start}}_t = \max(-\Delta B_{\overline{t-1}},\ S^{\min}_t)$$

8.12 Temperature

Beanstalk regularly crosses the price of $\not\Diamond 1$ over its value peg during long run decreases and increases in demand for Beans primarily by adjusting the *Temperature* at the beginning of each *Season* in an attempt to maintain an optimal state, or to move from its current state into an optimal state.

The *Temperature* change at the beginning of t is determined by R^D_{t-1} and the current state of Beanstalk with respect to ideal equilibrium. When R^D_{t-1} is excessively high or low, Beanstalk changes the *Temperature* more aggressively.

When $R^{D^{\text{upper}}} \leq R^D_{t-1}$ (*i.e.*, the debt level was excessively high):

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

When $R^{D^*} \leq R^D_{t-1} < R^{D^{\text{upper}}}$ (*i.e.*, the debt level was reasonably high):

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

When $R^{D^{\text{lower}}} \leq R^D_{t-1} < R^{D^*}$ (*i.e.*, the debt level was reasonably low):

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

When $R^D_{t-1} < R^{D^{\text{lower}}}$ (*i.e.*, the debt level was excessively low):

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

Thus, Beanstalk changes the *Temperature* to regularly cross the price of $\not{0}1$ over its value peg during long run decreases and increases in demand for Beans.

Current State: Temperature Changes	R^D_{t-1}: Excessively Low Debt	Reasonably Low Debt	Reasonably High Debt	Excessively High Debt
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	0
Steady Toward	1	1	-1	-1
Accelerating Toward	3	3	-3	-3

Figure 12: Temperature Changes From Current State and R^D_{t-1}

$P_{\overline{t-1}}$ & Demand Changes: Temperature Changes		R^D_{t-1}: Excessively Low Debt	Reasonably Low Debt	Reasonably High Debt	Excessively High Debt
$P_{\overline{t-1}} > 1$	**Increasing**	-3	-3	-3	-3
	Steady	-3	-3	-1	-1
	Decreasing	-1	-1	0	0
$P_{\overline{t-1}} < 1$	**Increasing**	0	0	1	1
	Steady	1	1	3	3
	Decreasing	3	3	3	3

Figure 13: Temperature Changes From $P_{\overline{t-1}}$, Demand for Soil Changes and R^D_{t-1}

8.13 Flood

Beanstalk sells newly minted Beans on the open market during long run increases in demand for Beans when increasing the Bean supply and lowering the *Temperature* has not crossed the average nor current prices of $\mathbb{B}1$ over its value peg at the end of a *Season*.

If $V < P_{\overline{t-1}}$, it is *Raining*. If it is *Raining* and $R^D_{t-1} < R^{D^{\text{lower}}}$, the *Farm* is *Oversaturated*. If the *Farm* is *Oversaturated* for ξ consecutive *Seasons* or more, each *Season* in which it continues to be *Oversaturated* there is a *Flood*. At the beginning of each *Season* during a *Flood*, Beanstalk returns the price of $\mathbb{B}1$ in each liquidity pool on the *Oracle Whitelist* to its value peg by minting additional Beans and selling them directly in the pools. Proceeds from the sale are distributed to *Stalkholders* at the beginning of t in proportion to their *Stalk* holdings when the *Farm* became *Oversaturated*. At the beginning of a *Flood*, all *Pods* that grew from Beans *Sown* before the *Farm* became *Oversaturated Ripen* and become *Harvestable*.

The number of Beans that are minted and sold to return the price of $\mathbb{B}1$ to its value peg (ΔB_{t-1}), such that $\Delta B_{t-1} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, is calculated from the sum of differences between the optimal number of Beans and the number of Beans in each $\mathbb{B}$:y liquidity pool on the *Oracle Whitelist* at the end of the previous *Season*.

In a *Flood*, m_t for a given number of *Unharvestable Pods* that grew prior to the the *Farm* becoming *Oversaturated* (D_γ), such that $D_\gamma \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, a_t, $\Delta B_{\overline{t-1}}$, and ΔB_{t-1} is:

$$m_t = D_\gamma + a_t + \Delta B_{\overline{t-1}} + \Delta B_{t-1}$$

Thus, Beanstalk regularly crosses the price of $\not\Beta 1$ over its value peg during both long run increases and decreases in demand for Beans.

9 Market

Current DEXs are unable to attract liquidity without offering protocol-native emissions derived primarily from AMM trading fees. Beanstalk's ability to attract liquidity without fee-based emissions allows it to create a DEX without trading fees. The *Market* is the Beanstalk-native DEX. Specifications of the *Market* are outside the scope of this whitepaper. For information on the *Market*, refer to the Appendix.

10 Depot

Current complex interactions with Ethereum-native protocols are tedious, cumbersome and expensive. The *Depot* facilitates complex, gas-efficient interactions with other Ethereum-native protocols in a single transaction. Any protocol with a *Pipeline* to the *Depot* can be used via Beanstalk in a single transaction. *Pipelines* to the *Depot* can be added via Beanstalk governance. The specifications of specific *Pipelines* are outside the scope of this whitepaper. For information on the *Depot*, refer to the Appendix.

11 Economics

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

11.1 Ownership Concentration

A design that lowers the Gini coefficient[35] of Beans and *Stalk* over time is essential to censorship resistance.

Older *Deposits* have their *Stalk* from *Seeds* diluted relative to newer *Deposits* every *Season*. Therefore, newly minted Beans are more widely distributed over time.

Beanstalk does not require a pre-mine. The first 100 Beans are created when the `init` function is called to deploy Beanstalk.

11.2 Strong Credit

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

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

[35] wikipedia.org/wiki/Gini_coefficient

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

11.3 Marginal Rate of Substitution

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

11.4 Low Friction

Minimizing the cost of using Beans and barriers to the *Farm* maximize utility for users and appeal to creditors. The *Depot* realizes the full benefits of composability on Ethereum.

The FIFO *Pod Harvest* schedule allows smaller *Sowers* to participate in peg maintenance and decreases the benefit of large scale price manipulation. The combination of non-expiry, the FIFO *Harvest* schedule, transferability and a liquid secondary market (see Appendix) enables *Sowers* to *Sow* Beans as efficiently as possible. By maximizing the efficiency of the *Soil* market, Beanstalk minimizes its cost to attract creditors, the durations and magnitudes of price deviations below its value peg, and excess *Pod* issuance.

11.5 Equilibrium

Equilibrium is a state of equivalent marginal quantity supplied and demanded. Beanstalk affects the supply of and demand for Beans to regularly cross the equilibrium price of $\beta 1$ over its value peg.

While Beanstalk can arbitrarily increase the Bean supply when the equilibrium price of $\beta 1$ is above its value peg, Beanstalk cannot arbitrarily decrease the Bean supply when the equilibrium price of $\beta 1$ 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* and Beans are functions of the *Temperature* and the Bean price. By changing the *Temperature*, Beanstalk affects decreases in the Bean supply and changes in demand for Beans.

11.6 Incentives

Beanstalk-native financial incentives consistently increase trustlessness, stability and liquidity over time by coordinating independently financially motivated actors (*i.e*, *Stalkholders* and *Sowers*).

The *Stalk System* incentivizes (1) leaving assets *Deposited* in the *Silo* continuously by creating opportunity cost to *Withdraw* assets from the *Silo*, (2) adding value to liquidity pools with Beans by rewarding more *Seeds* to *Deposited* LP tokens than *Deposited* β, and (3) returning the price of $\beta 1$ to its value peg by allowing *Conversions* within the *Silo* without forfeiting *Stalk*.

Beanstalk is governed by *Stalkholders*. Anyone with *Stalk* stands to profit from future growth of Beanstalk, but are not owed anything by Beanstalk.

When $P_{\bar{t}} < V$, there is an incentive to *Withdraw* assets from the *Silo*. The combination of the *Stalk System* and *Withdrawal Freeze* reduces this incentive significantly.

When $V < P_{\bar{t}}$, there is an incentive to buy Beans to earn a portion of the upcoming Bean seigniorage. This is exacerbated when R^D is lower. The combination of the commitment to automatically return the price of $\textcircled{1}$ to its value peg and distribute proceeds from the sale to current *Stalkholders* based on *Stalk* ownership when the *Farm* became *Oversaturated*, and the *Withdrawal Freeze*, removes this incentive entirely during *Seasons* where R^D_{t-1} is excessively low, and reduces it significantly otherwise.

Thus, Beanstalk consistently increases trustlessness, stability and liquidity over time.

12 Risk

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

The Beanstalk code base and peg maintenance mechanism are novel. Neither had been tested in the "real world" prior to the initial Beanstalk deployment. Portions of the Beanstalk code base are unaudited.[36,37,38] 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 composability and fungible token standards, (2) a DEX protocol that runs on (1), and (3) a non-network-native exogenous value convertible stablecoin protocol native to (1) that offers convertibility to the non-network-native exogenous value and trades on (2).

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. Curve is one of the largest Ethereum-native DEX protocols by volume.[39] USDC and USDT are the largest non-network-native exogenous value USD stablecoin protocols, and DAI is the largest network-native exogenous value USD stablecoin protocol (by market capitalization). In general, open source protocols with large amounts of value on them (*e.g.*, Ethereum, Curve, USDC, USDT and DAI) are high value targets for exploits. Long track records indicate security. We assume the security of the Ethereum blockchain, ERC-20 Standard, Curve, USDC, USDT and DAI.

The Beanstalk price oracle contains exposure to risk related to the underlying collateral of x (*e.g.*, USDC, USDT, DAI). There is no guarantee the centralized operators of USDC and USDT hold non-network-native exogenous value worth at least 100% of all outstanding protocol liabilities or will not ban them from the 3CRV pool, although either would cause significant financial self-harm. The operators of USDC and USDT may alter their convertibility policies, which would negatively affect their respective stablecoins as price sources for USD, and thereby corrupt the accuracy of 3CRV as a price source for USD. However, in theory, if the price of x falls below V, it 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.

13 Future Work

Beanstalk is a work in progress. The following are potential improvements that can be incorporated into Beanstalk as one or more *BIPs*:

[36] bean.money/07-13-22-halborn-report
[37] bean.money/07-22-22-tob-report
[38] bean.money/07-22-22-tob-fix-review
[39] defiprime.com/dex-volume

- Permissionless governance can be reimplemented.
- *Stalk* can become liquid to further increase composability and decrease friction.
- Beanstalk can distribute yield received from other protocols by *Deposited* assets to its *Depositor*.
- The *Silo* can support additional token standards.
- The *Withdrawal Freeze* can be removed.
- The decentralized price oracle is unlikely to remain sufficiently manipulation resistant at scale, and can be significantly improved.
- The calculation of $\Delta B_{\overline{t-1}}$ can account for inaccuracies in the calculation due to frictions (*e.g*, AMM trading fees).
- Additional x and Ethereum-native DEXs can be incorporated into $P_{\overline{t-1}}$.
- 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.
- The *Market* can be further developed.
- The *Depot* can support additional *Pipelines*.
- Beanstalk can issue unique assets with different value pegs on Ethereum.

14 Appendix

14.1 Current Parameters

The following are the current parameters of Beanstalk:

- $\xi = 0$;
- $K^{\min} = 0.1\%$;
- $w_1 = 1$;
- $R^{S^{\min}} = 0.1\%$;
- $R^{S^{\max}} = 25\%$;
- $R^{D^{\text{lower}}} = 5\%$;
- $R^{D^*} = 15\%$;
- $R^{D^{\text{upper}}} = 25\%$;
- $\frac{\partial \Delta S}{\partial t}^{\text{lower}} = 95\%$;
- $\frac{\partial \Delta S}{\partial t}^{\text{upper}} = 105\%$;
- $\frac{\partial R^S}{\partial t}^{\text{upper}} = 105\%$;
- $\Delta E_t^{u^{\text{lower}}} = -60$; and
- $\Delta E_t^{u^{\text{upper}}} = 60$.

14.2 Deposit Whitelist

The following ERC-20 Standard tokens are *Whitelisted* for *Deposit* in the *Silo*:

14.2.1 $\cancel{\$}$

1. **Token Address:** The $\cancel{\$}$ token address is 0xBEA0000029AD1c77D3d5D23Ba2D8893dB9d1Efab.
2. **BDV Function:** The BDV of 1 $\cancel{\$}$ is 1 $\cancel{\$}$.
   We define $f^{\cancel{\$}}(z^{\cancel{\$}})$ as:
   $$f^{\cancel{\$}}(z^{\cancel{\$}}) = z^{\cancel{\$}}$$
3. **Stalk per BDV:** $\cancel{\$}$ *Deposits* receive 1 Stalk per BDV upon *Deposit* (*i.e.*, $k^{\cancel{\$}} = 1$).
4. **Seed per BDV:** $\cancel{\$}$ *Deposits* receive 2 Seeds per BDV upon *Deposit* (*i.e.*, $c^{\cancel{\$}} = 2$).

14.2.2 Φ

1. **Token Address:** The Φ token address is 0xc9C32cd16Bf7eFB85Ff14e0c8603cc90F6F2eE49.
2. **BDV Function:** The BDV of Φ is calculated using the number of Beans ($\Phi^{\cancel{\$}}_{\Xi-1}$), such that $\Phi^{\cancel{\$}}_{\Xi-1} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, and number of 3CRV ($\Phi^{3\text{CRV}}_{\Xi-1}$) in the BEAN:3CRV Curve pool at the end of the last block, the 3CRV virtual price ($P^{3\text{CRV}}$), the A parameter of the pool (Φ^A), such that $\Phi^A \in \{j \times 10^{-2} \mid j \in \mathbb{Z}^+\}$, and the Φ virtual price (P^Φ), such that $\Phi^{3\text{CRV}}_{\Xi-1},\ P^{3\text{CRV}},\ P^\Phi \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$.

   Beanstalk calculates a flash-loan-resistant price invariant for the BEAN:3CRV Curve pool ($\zeta^\Phi_{\Xi-1}$), such that $\zeta^\Phi_{\Xi-1} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, by calling the Curve[40] `get_D` function on $\Phi^{\cancel{\$}}_{\Xi-1}$, $\Phi^{3\text{CRV}}_{\Xi-1}$, $P^{3\text{CRV}}$ and Φ^A as:
   $$\zeta^\Phi_{\Xi-1} = \texttt{get\_D}([\Phi^{\cancel{\$}}_{\Xi-1},\ \Phi^{3\text{CRV}}_{\Xi-1} \times P^{3\text{CRV}}],\ \Phi^A)$$
   Beanstalk calculates a flash-loan-resistant total number of Φ ($\Phi_{\Xi-1}$), such that $\Phi_{\Xi-1} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, from $\zeta^\Phi_{\Xi-1}$ and P^Φ as:
   $$\Phi_{\Xi-1} = \frac{\zeta^\Phi_{\Xi-1}}{P^\Phi}$$
   Beanstalk calculates the flash-loan-resistant USD price of $\cancel{\$}1$ from the BEAN:3CRV Curve pool ($\$^{\cancel{\$}(\Phi)}_{\Xi-1}$), such that $\$^{\cancel{\$}(\Phi)}_{\Xi-1} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, by calling the Curve `get_y` function on $\Phi^{\cancel{\$}}_{\Xi-1}$, $\Phi^{3\text{CRV}}_{\Xi-1}$ and $P^{3\text{CRV}}$ as:
   $$\$^{\cancel{\$}(\Phi)}_{\Xi-1} = \Phi^{\cancel{\$}}_{\Xi-1} - \texttt{get\_y}(0, 1, \Phi^{\cancel{\$}}_{\Xi-1} + 1, [\Phi^{\cancel{\$}}_{\Xi-1},\ \Phi^{3\text{CRV}}_{\Xi-1} \times P^{3\text{CRV}}]) - 10^{-6}$$
   Beanstalk calculates the BDV of 3CRV $f^{3\text{CRV}}(z^{3\text{CRV}})$ from $\$^{\cancel{\$}(\Phi)}_{\Xi-1}$ and $P^{3\text{CRV}}$ as:
   $$f^{3\text{CRV}}(z^{3\text{CRV}}) = \frac{z^{3\text{CRV}} \times P^{3\text{CRV}}}{\$^{\cancel{\$}(\Phi)}_{\Xi-1}}$$
   We define $f^\Phi(z^\Phi)$ for a given $\Phi^{\cancel{\$}}_{\Xi-1}$, $f^{3\text{CRV}}(z^{3\text{CRV}})$, $\Phi^{3\text{CRV}}_{\Xi-1}$ and $\Phi_{\Xi-1}$ as:
   $$f^\Phi(z^\Phi) = \frac{z^\Phi \times (\Phi^{\cancel{\$}}_{\Xi-1} + f^{3\text{CRV}}(\Phi^{3\text{CRV}}_{\Xi-1}))}{\Phi_{\Xi-1}}$$
3. **Stalk per BDV:** Φ *Deposits* receive 1 Stalk per BDV upon *Deposit* (*i.e.*, $k^\Phi = 1$).
4. **Seed per BDV:** Φ *Deposits* receive 4 Seeds per BDV upon *Deposit* (*i.e.*, $c^\Phi = 4$).

[40] etherscan.io/address/0xc9C32cd16Bf7eFB85Ff14e0c8603cc90F6F2eE49#code

14.2.3 $\mathfrak{z}^{\emptyset}$

1. **Token Address:** The $\mathfrak{z}^{\emptyset}$ token address is 0x1BEA0050E63e05FBb5D8BA2f10cf5800B6224449.
2. **BDV Function:** The BDV of $\mathfrak{z}^{\emptyset}$ is calculated using $f^{\emptyset}(z^{\emptyset})$, $\mathfrak{L}^{\emptyset}$ and $\mathfrak{Z}^{\emptyset}$.
   We define $f^{\mathfrak{z}^{\emptyset}}(z^{\mathfrak{z}^{\emptyset}})$ as:
   $$f^{\mathfrak{z}^{\emptyset}}(z^{\mathfrak{z}^{\emptyset}}) = f^{\emptyset}\left(\frac{z^{\mathfrak{z}^{\emptyset}} \times \mathfrak{L}^{\emptyset}}{\mathfrak{Z}^{\emptyset}}\right)$$
3. **Stalk per BDV:** $\emptyset$ *Deposits* receive 1 Stalk per BDV upon *Deposit* (*i.e.*, $k^{\mathfrak{z}^{\emptyset}} = 1$).
4. **Seed per BDV:** $\emptyset$ *Deposits* receive 2 Seeds per BDV upon *Deposit* (*i.e.*, $c^{\mathfrak{z}^{\emptyset}} = 2$).

14.2.4 $\mathfrak{z}^{\Phi}$

1. **Token Address:** The $\mathfrak{z}^{\Phi}$ token address is 0x1BEA3CcD22F4EBd3d37d731BA31Eeca95713716D.
2. **BDV Function:** The BDV of $\mathfrak{z}^{\Phi}$ is calculated using $f^{\Phi}(z^{\Phi})$, $\mathfrak{L}^{\Phi}$ and $\mathfrak{Z}^{\Phi}$.
   We define $f^{\mathfrak{z}^{\Phi}}(z^{\mathfrak{z}^{\Phi}})$ as:
   $$f^{\mathfrak{z}^{\Phi}}(z^{\mathfrak{z}^{\Phi}}) = f^{\Phi}\left(\frac{z^{\mathfrak{z}^{\Phi}} \times \mathfrak{L}^{\Phi}}{\mathfrak{Z}^{\Phi}}\right)$$
3. **Stalk per BDV:** $\mathfrak{z}^{\Phi}$ *Deposits* receive 1 Stalk per BDV upon *Deposit* (*i.e.*, $k^{\mathfrak{z}^{\Phi}} = 1$).
4. **Seed per BDV:** $\mathfrak{z}^{\Phi}$ *Deposits* receive 4 Seeds per BDV upon *Deposit* (*i.e.*, $c^{\mathfrak{z}^{\Phi}} = 4$).

14.3 Former Governance

The following has been removed from Section 5.5 Governance as part of the updates to reflect Beanstalk's current permissioned governance system and is left here to contribute to the discussion around a future permissionless governance system.

The submitter of a *BIP* automatically votes in favor of the *BIP*, cannot rescind their vote, and cannot have less than $K^{\min}$ of total outstanding *Stalk* after an interaction with the *Silo*, until the end of the *Voting Period*.

When a *BIP* passes or has a two-thirds majority, it must be manually committed to the Ethereum blockchain. To encourage prompt commitment of *BIPs* 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 six seconds that elapse past the end of its *Voting Period* (E_{BIP}) for 1,800 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:

$$a^q = 100 \times 1.01^{\min\left\{\left\lfloor \frac{E_q - E_{\text{BIP}}}{6} \right\rfloor,\ 300\right\}}$$

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 committed with a two-thirds supermajority before the end of its *Voting Period*, $a^q = 100$.

14.4 Convert Whitelist

The following *Conversions* within the *Silo* are *Whitelisted*:

14.4.1 $\lambda \rightarrow \lambda$

1. **From Token Address:** The from token address must match the to token address.
2. **To Token Address:** The to token address must match the from token address.
3. **Conditions:** *Deposited* λ can be *Converted* to a λ *Deposit* at anytime.
4. **Convert Function:** The number of λ received for *Converting Deposited* λ within the *Silo* is equivalent to the number of λ *Converted*. Therefore, we define function as:

$$f^{\lambda \rightarrow \lambda}(z^{\lambda}) = z^{\lambda}$$

14.4.2 $\emptyset \rightarrow \Phi$

1. **From Token Address:** The $\emptyset$ token address is 0xBEA0000029AD1c77D3d5D23Ba2D8893dB9d1Efab.
2. **To Token Address:** The Φ token address is 0xc9C32cd16Bf7eFB85Ff14e0c8603cc90F6F2eE49.
3. **Conditions:** *Deposited* $\emptyset$ cannot be *Converted* to *Deposited* Φ when the USD price of $\emptyset 1$ in the pool ($\$^{\emptyset(\Phi)}$), such that $\$^{\emptyset(\Phi)} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$, is below \$1 (*i.e*, $\$^{\emptyset(\Phi)} < 1$).

   $\$^{\emptyset(\Phi)}$ is calculated using the number of Beans ($\Phi^{\emptyset}$), such that $\Phi^{\emptyset} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$, and number of 3CRV ($\Phi^{\text{3CRV}}$), such that $\Phi^{\text{3CRV}} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^{+}\}$, in the BEAN:3CRV Curve pool, P^{3CRV}, Φ^{A}, P^{Φ}.

   Beanstalk calculates a price invariant for the BEAN:3CRV Curve pool (ζ^{Φ}), such that $\zeta^{\Phi} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^{+}\}$, by calling the Curve `get_D` function on $\Phi^{\emptyset}$, Φ^{3CRV}, P^{3CRV} and Φ^{A} as:

$$\zeta^{\Phi} = \texttt{get\_D}([\Phi^{\emptyset},\ \Phi^{\text{3CRV}} \times P^{\text{3CRV}}],\ \Phi^{A})$$

   Beanstalk calculates a total number of Φ, such that $\Phi \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^{+}\}$, from ζ^{Φ} and P^{Φ} as:

$$\Phi = \frac{\zeta^{\Phi}}{P^{\Phi}}$$

   Beanstalk calculates the $\$^{\emptyset(\Phi)}$ by calling the Curve `get_y` function on $\Phi^{\emptyset}$, Φ^{3CRV} and P^{3CRV} as:

$$\$^{\emptyset(\Phi)} = \Phi^{\emptyset} - \texttt{get\_y}(0, 1, \Phi^{\emptyset} + 1, [\Phi^{\emptyset},\ \Phi^{\text{3CRV}} \times P^{\text{3CRV}}]) - 10^{-6}$$

4. **Convert Function:** The number of Φ received for *Converting Deposited* Beans within the *Silo* for a given minimum Φ received ($\Phi^{\min}$), such that $\Phi^{\min} \in \{j \times 10^{-18} \mid j \in \mathbb{N}\}$, is the result of calling the Curve `add_liquidity` function on Φ and $\Phi^{\min}$ as:

$$f^{\emptyset \rightarrow \Phi}(z^{\emptyset}) = \texttt{add\_liquidity}(\Phi,\ [z^{\emptyset},\ 0],\ \Phi^{\min})$$

14.4.3 $\Phi \rightarrow \emptyset$

1. **From Token Address:** The Φ token address is 0xc9C32cd16Bf7eFB85Ff14e0c8603cc90F6F2eE49.
2. **To Token Address:** The $\emptyset$ token address is 0xBEA0000029AD1c77D3d5D23Ba2D8893dB9d1Efab.

3. **Conditions:** *Deposited* Φ cannot be *Converted* to *Deposited* $\not\Phi$ when the price of $\not\Phi 1$ in the pool is greater than or equal to \$1 (*i.e*, $1 \leq \$^{\not\Phi(\Phi)}$).

4. **Convert Function:** The number of Beans received for *Converting Deposited* Φ within the *Silo* for a given minimum Beans received ($\not\Phi^{\min}$), such that $\not\Phi^{\min} \in \{j \times 10^{-18} \mid j \in \mathbb{N}\}$, is the result of calling the Curve `remove_liquidity_one_coin` function on Φ and $\not\Phi^{\min}$ as:

$$f^{\Phi \to \not\Phi}(z^{\Phi}) = \texttt{remove\_liquidity\_one\_coin}(\Phi,\ z^{\Phi},\ 0,\ \not\Phi^{\min})$$

14.4.4 $\mathfrak{z}^{\not\Phi} \to \mathfrak{z}^{\Phi}$

1. **From Token Address:** The $\mathfrak{z}^{\not\Phi}$ token address is 0x1BEA0050E63e05FBb5D8BA2f10cf5800B6224449.

2. **To Token Address:** The $\mathfrak{z}^{\Phi}$ token address is 0x1BEA3CcD22F4EBd3d37d731BA31Eeca95713716D.

3. **Conditions:** *Deposited* $\mathfrak{z}^{\not\Phi}$ cannot be *Converted* to *Deposited* $\mathfrak{z}^{\Phi}$ when the price of $\not\Phi 1$ in the BEAN:3CRV liquidity pool is less than or equal to \$1 (*i.e*, $\$^{\not\Phi(\Phi)} \leq 1$).

4. **Convert Function:** The number of $\mathfrak{z}^{\Phi}$ received for *Converting Deposited* $\mathfrak{z}^{\not\Phi}$ within the *Silo* for a given $\mathfrak{S}$, $\mathfrak{Z}^{\Phi}_{\bigotimes}$, $\mathfrak{Z}^{\Phi}$, $\mathfrak{R}^{\Phi}$, $\mathfrak{R}^{\not\Phi}$, $\mathfrak{Z}^{\not\Phi}$ and minimum *Unripe* Φ received ($\mathfrak{z}^{\Phi^{\min}}$), such that $\mathfrak{z}^{\Phi^{\min}} \in \{j \times 10^{-18} \mid j \in \mathbb{N}\}$, is the result of calling the Curve `add_liquidity` function as:

$$f^{\mathfrak{z}^{\not\Phi} \to \mathfrak{z}^{\Phi}}(z^{\mathfrak{z}^{\not\Phi}}) = \frac{\mathfrak{S} \times \mathfrak{Z}^{\Phi}_{\bigotimes} \times \mathfrak{Z}^{\Phi}}{7.7 \times 10^{7} \times \mathfrak{R}^{\Phi}} \times \texttt{add\_liquidity}\left(\Phi,\ \left[\frac{z^{\mathfrak{z}^{\not\Phi}} \times \mathfrak{R}^{\not\Phi}}{\mathfrak{Z}^{\not\Phi}},\ 0\right],\ \mathfrak{z}^{\Phi^{\min}}\right)$$

14.4.5 $\mathfrak{z}^{\Phi} \to \mathfrak{z}^{\not\Phi}$

1. **From Token Address:** The $\mathfrak{z}^{\Phi}$ token address is 0x1BEA3CcD22F4EBd3d37d731BA31Eeca95713716D.

2. **To Token Address:** The $\mathfrak{z}^{\not\Phi}$ token address is 0x1BEA0050E63e05FBb5D8BA2f10cf5800B6224449.

3. **Conditions:** *Deposited* $\mathfrak{z}^{\Phi}$ cannot be *Converted* to *Deposited* $\mathfrak{z}^{\not\Phi}$ when the price of $\not\Phi 1$ in the pool is greater than or equal \$1 (*i.e*, $1 \leq \$^{\not\Phi(\Phi)}$).

4. **Convert Function:** The number of $\mathfrak{z}^{\not\Phi}$ received for *Converting Deposited* $\mathfrak{z}^{\Phi}$ within the *Silo* for a given $\mathfrak{Z}^{\Phi}$, $\mathfrak{S}$, $\mathfrak{Z}^{\Phi}_{\bigotimes}$, $\mathfrak{R}^{\Phi}$ and minimum *Unripe* Beans received ($\mathfrak{z}^{\not\Phi^{\min}}$), such that $\mathfrak{z}^{\not\Phi^{\min}} \in \{j \times 10^{-18} \mid j \in \mathbb{N}\}$, is the result of calling the Curve `remove_liquidity_one_coin` function as:

$$f^{\mathfrak{z}^{\Phi} \to \mathfrak{z}^{\not\Phi}}(z^{\mathfrak{z}^{\Phi}}) = \frac{7.7 \times 10^{7} \times \mathfrak{Z}^{\Phi}}{\mathfrak{S} \times \mathfrak{Z}^{\Phi}_{\bigotimes}} \times \texttt{remove\_liquidity\_one\_coin}\left(\Phi,\ \left[\frac{z^{\mathfrak{z}^{\Phi}} \times \mathfrak{R}^{\Phi}}{\mathfrak{Z}^{\Phi}},\ 0\right],\ \mathfrak{z}^{\not\Phi^{\min}}\right)$$

14.5 Barn

The following ERC-20 Standard tokens were *Whitelisted* for *Deposit* in the *Silo* at the end of the block prior to the *Exploit*. Upon *Replant*, *Stalkholders* at the end of the block prior to the *Exploit* received *Stalk* and *Seeds* based on their *Deposits* at the end of the block prior to the *Exploit*. All non-Bean *Deposits* are credited with 4 *Seeds* per BDV upon *Deposit*, independent of c^{λ}. The previous c^{λ}, total supply and BDV of each token at the end of the block prior to the *Exploit* have been included for reference.

14.5.1 Old $\emptyset$

1. **Token Address:** The old $\emptyset$ token address is 0xDC59ac4FeFa32293A95889Dc396682858d52e5Db.
2. **BDV Function:** The BDV of 1 $\emptyset$ is 1 $\emptyset$.
   Therefore, we defined $f^{\emptyset}(z^{\emptyset})$ as:
   $$f^{\emptyset}(z^{\emptyset}) = z^{\emptyset}$$
3. **Stalk per BDV:** $\emptyset$ *Deposits* received 1 Stalk per BDV upon *Deposit* (*i.e.*, $k^{\emptyset} = 1$).
4. **Seed per BDV:** $\emptyset$ *Deposits* received 2 Seeds per BDV upon *Deposit* (*i.e.*, $c^{\emptyset} = 2$).
5. **Total Supply:** There were 108155457.359439 old $\emptyset$ at the end of the block prior to the *Exploit*.
6. **BDV Per Token:** The BDV per old $\emptyset$ at the end of the block prior to the *Exploit* was 1.

14.5.2 Old BEAN:ETH Uniswap V2 LP Tokens ($\beth$)

1. **Token Address:** The $\beth$ token address is 0x87898263b6c5babe34b4ec53f22d98430b91e371.
2. **BDV Function:** The BDV of $\beth$ was calculated using the last traded price in the BEAN:ETH Uniswap v2 pool unless there was an interaction with the pool in the current block. The last traded price was a function of the current number of Beans in the pool ($\beth^{\emptyset}$), such that $\beth^{\emptyset} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$. If there was an interaction with the pool in the current block, Beanstalk used the time weighted average number of Beans in the pool from the start of the current *Season* to the current block ($\beth_t^{\emptyset}$), such that $\beth_t^{\emptyset} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^{+}\}$ unless the `sunrise` function was also called in the current block. If there was an interaction with the pool and the `sunrise` function was called in the current block, $\beth$ *Deposits* are not accepted.
   Therefore, we defined $f^{\beth}(z^{\beth})$ for a given timestamp of the last interaction with the pool ($E_{\beth}$), current block timestamp (E_{Ξ}^{*}), E_t, $\beth_t^{\emptyset}$, the current total number of $\beth$ in the current block ($\beth$), such that $\beth \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^{+}\}$, and $\beth^{\emptyset}$ as:
   $$f^{\beth}(z^{\beth}) = \begin{cases} \text{FAIL} & \text{if } E_{\beth} = E_{\Xi}^{*} \;\&\; E_{\Xi}^{*} = E_t \\ \dfrac{z^{\beth} \times 2 \times \beth_t^{\emptyset}}{\beth} & \text{if } E_{\beth} = E_{\Xi}^{*} \\ \dfrac{z^{\beth} \times 2 \times \beth^{\emptyset}}{\beth} & \text{else} \end{cases}$$
3. **Stalk per BDV:** $\beth$ *Deposits* received 1 Stalk per BDV upon *Deposit* (*i.e.*, $k^{\beth} = 1$).
4. **Seed per BDV:** $\beth$ *Deposits* received 4 Seeds per BDV upon *Deposit* (*i.e.*, $c^{\beth} = 4$).
5. **Total Supply:** There were 0.540894218294675521 $\beth$ at the end of the block prior to the *Exploit*.
6. **BDV Per Token:** The BDV per $\beth$ at the end of the block prior to the *Exploit* was 119,894,802.186829.

14.5.3 Old BEAN:3CRV Curve LP Tokens (⅂)

1. **Token Address:** The ⅂ token address is 0x3a70DfA7d2262988064A2D051dd47521E43c9BdD.

2. **BDV Function:** The BDV of ⅂ was calculated using the number of Beans ($\urcorner^{\emptyset}_{\Xi-1}$), such that $\urcorner^{\emptyset}_{\Xi-1} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, and number of 3CRV ($\urcorner^{3\text{CRV}}_{\Xi-1}$) in the old BEAN:3CRV Curve pool at the end of the last block, $P^{3\text{CRV}}$, the A parameter of the pool ($\urcorner^A$), such that $\urcorner^A \in \{j \times 10^{-2} \mid j \in \mathbb{Z}^+\}$, and the ⅂ virtual price ($P^{\urcorner}$), such that $\urcorner^{3\text{CRV}}_{\Xi-1},\ P^{\urcorner} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$.

   Beanstalk calculated a flash-loan-resistant price invariant for the BEAN:3CRV Curve pool ($\zeta^{\urcorner}$), such that $\zeta^{\urcorner} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, by calling the Curve `get_D` function on $\urcorner^{\emptyset}_{\Xi-1}$, $\urcorner^{3\text{CRV}}_{\Xi-1}$, $P^{3\text{CRV}}$ and $\urcorner^A$ as:

$$\zeta^{\urcorner} = \texttt{get\_D}([\urcorner^{\emptyset}_{\Xi-1},\ \urcorner^{3\text{CRV}}_{\Xi-1} \times P^{3\text{CRV}}],\ \urcorner^A)$$

   Beanstalk calculated a flash-loan-resistant total number of ⅂ ($\urcorner_{\Xi-1}$), such that $\urcorner_{\Xi-1} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, from $\zeta^{\urcorner}$ and $P^{\urcorner}$ as:

$$\urcorner_{\Xi-1} = \frac{\zeta^{\urcorner}}{P^{\urcorner}}$$

   Beanstalk calculated the USD price of $\emptyset 1$ from the BEAN:3CRV Curve pool ($\$^{\emptyset(\urcorner)}$), such that $\$^{\emptyset(\urcorner)} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, by calling the Curve `get_y` function on $\urcorner^{\emptyset}_{\Xi-1}$, $\urcorner^{3\text{CRV}}_{\Xi-1}$ and $P^{3\text{CRV}}$ as:

$$\$^{\emptyset(\urcorner)} = \urcorner^{\emptyset}_{\Xi-1} - \texttt{get\_y}(0, 1, \urcorner^{\emptyset}_{\Xi-1} + 1, [\urcorner^{\emptyset}_{\Xi-1},\ \urcorner^{3\text{CRV}}_{\Xi-1} \times P^{3\text{CRV}}]) - 10^{-6}$$

   Beanstalk calculated $f^{3\text{CRV}}(z^{3\text{CRV}})$ from $\$^{\emptyset(\urcorner)}$ and $P^{3\text{CRV}}$ as:

$$f^{3\text{CRV}}(z^{3\text{CRV}}) = \frac{z^{3\text{CRV}} \times P^{3\text{CRV}}}{\$^{\emptyset(\urcorner)}}$$

   We defined $f^{\urcorner}(z^{\urcorner})$ for a given $\urcorner^{\emptyset}_{\Xi-1}$, $f^{3\text{CRV}}(z^{3\text{CRV}})$, $\urcorner^{3\text{CRV}}_{\Xi-1}$ and $\urcorner_{\Xi-1}$ as:

$$f^{\urcorner}(z^{\urcorner}) = \frac{z^{\urcorner} \times (\urcorner^{\emptyset}_{\Xi-1} + f^{3\text{CRV}}(\urcorner^{3\text{CRV}}_{\Xi-1}))}{\urcorner_{\Xi-1}}$$

3. **Stalk per BDV:** ⅂ *Deposits* received 1 Stalk per BDV upon *Deposit* (*i.e.*, $k^{\urcorner} = 1$).

4. **Seed per BDV:** ⅂ *Deposits* received 4 Seeds per BDV upon

5. **Total Supply:** There were 79284313.822927052565331157 ⅂ at the end of the block prior to the *Exploit*.

6. **BDV Per Token:** The BDV per ⅂ at the end of the block prior to the *Exploit* was 0.992035.

14.5.4 Old BEAN:LUSD Curve LP Tokens (⅃)

1. **Token Address:** The ⅃ token address is 0xD652c40fBb3f06d6B58Cb9aa9CFF063eE63d465D.

2. **BDV Function:** The BDV of ⅃ was calculated using the number of LUSD ($\Omega^{\text{LUSD}}_{\Xi-1}$) and number of 3CRV ($\Omega^{3\text{CRV}}_{\Xi-1}$) in the LUSD:3CRV Curve pool (Ω) at the end of the last block, $P^{3\text{CRV}}$, the A parameter of the pool (Ω^A), such that $\Omega^A \in \{j \times 10^{-2} \mid j \in \mathbb{Z}^+\}$, $P^{3\text{CRV}}$ and $\$^{\emptyset(\urcorner)}$, the Ω virtual price (P^{Ω}), the ⅃ virtual price ($P^{\lrcorner}$), such that $\Omega^{\text{LUSD}}_{\Xi-1},\ \Omega^{3\text{CRV}}_{\Xi-1},\ P^{\Omega},\ P^{\lrcorner} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, $P^{3\text{CRV}}$ and $\$^{\emptyset(\urcorner)}$.

   Beanstalk calculated a flash-loan-resistant price invariant for the LUSD:3CRV Curve pool (ζ^{Ω}), such that $\zeta^{\Omega} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, by calling the Curve `get_D` function on $\Omega^{\text{LUSD}}_{\Xi-1}$, $\Omega^{3\text{CRV}}_{\Xi-1}$, $P^{3\text{CRV}}$ and Ω^A as:

$$\zeta^{\Omega} = \texttt{get\_D}([\Omega^{\text{LUSD}}_{\Xi-1},\ \Omega^{3\text{CRV}}_{\Xi-1} \times P^{3\text{CRV}}],\ \Omega^A)$$

Beanstalk calculated a flash-loan-resistant total number of Ω ($\Omega_{\Xi-1}$), such that $\Omega_{\Xi-1} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, from ζ^{Ω} and P^{Ω} as:

$$\Omega_{\Xi-1} = \frac{\zeta^{\Omega}}{P^{\Omega}}$$

Beanstalk calculated the USD price of 1 LUSD from the LUSD:3CRV Curve pool ($\$^{\text{LUSD}(\Omega))}$), such that $\$^{\text{LUSD}(\Omega))} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, by calling the Curve `get_y` function on $\Omega_{\Xi-1}^{\text{LUSD}}$, $\Omega_{\Xi-1}^{\text{3CRV}}$ and P^{3CRV} as:

$$\$^{\text{LUSD}(\Omega))} = (\Omega_{\Xi-1}^{\text{LUSD}} - \texttt{get\_y}(0, 1, \Omega_{\Xi-1}^{\text{LUSD}} + 1, [\Omega_{\Xi-1}^{\text{LUSD}},\ \Omega_{\Xi-1}^{\text{3CRV}} \times P^{\text{3CRV}}]) - 10^{-6})$$

We defined $f^{\beth}(z^{\beth})$ for a given $P^{\beth}$, $\$^{\text{LUSD}(\Omega))}$ and $\$^{\emptyset(\urcorner)}$ as:

$$f^{\beth}(z^{\beth}) = z^{\beth} \times P^{\beth} \times \min\left(1, \frac{\$^{\text{LUSD}(\Omega))}}{\$^{\emptyset(\urcorner)}}\right)$$

3. **Stalk per BDV: $\beth$** *Deposits* received 1 Stalk per BDV upon *Deposit* (*i.e.*, $k^{\beth} = 1$).
4. **Seed per BDV: $\beth$** *Deposits* received 3 Seeds per BDV upon *Deposit* (*i.e.*, $c^{\beth} = 3$).
5. **Total Supply:** There were 1637956.191657208904972868 $\beth$ at the end of the block prior to the *Exploit*.
6. **BDV Per Token:** The BDV per $\beth$ at the end of the block prior to the *Exploit* was 0.983108.

14.6 Oracle Whitelist

The following liquidity pool is *Whitelisted* for inclusion in the calculation of $\Delta B_{\overline{t-1}}$:

14.6.1 Φ

1. **Pool Address:** The Φ liquidity pool address is 0xc9C32cd16Bf7eFB85Ff14e0c8603cc90F6F2eE49.

2. $\Delta b_{\overline{t-1}}$ **Calculation:** The liquidity and time weighted average shortage or excess of Beans in the BEAN:3CRV Curve liquidity pool over the previous *Season* ($\Delta b^{\Phi}_{\overline{t-1}}$) is calculated as the difference between the optimal liquidity and time weighted average number of Beans ($\Phi^{\emptyset^*}_{\overline{t-1}}$) and the liquidity and time weighted average number of Beans ($\Phi^{\emptyset}_{\overline{t-1}}$) in Φ over the previous *Season*, such that $\Delta b^{\Phi}_{\overline{t-1}},\ \Phi^{\emptyset^*}_{\overline{t-1}},\ \Phi^{\emptyset}_{\overline{t-1}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$. $\Phi^{\emptyset^*}_{\overline{t-1}}$ is calculated from $\Phi^{\emptyset}_{\overline{t-1}}$, the time weighted average number of 3CRV in Φ over the previous *Season* ($\Phi^{3\text{CRV}}_{\overline{t-1}}$), such that $\Phi^{3\text{CRV}}_{\overline{t-1}} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, $P^{3\text{CRV}}$ and Φ^A. The absolute value of $\Delta b_{\overline{t-1}}$ is at most 1% of B_{t-1}.

   Beanstalk calculates a liquidity and time weighted average price invariant for Φ over the previous *Season* ($\zeta^{\Phi}_{\overline{t-1}}$), such that $\zeta^{\Phi}_{\overline{t-1}} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, by calling the Curve `get_D` function on $\Phi^{\emptyset}_{\overline{t-1}}$, $\Phi^{3\text{CRV}}_{\overline{t-1}}$, $P^{3\text{CRV}}$ and Φ^A as:

$$\zeta^{\Phi}_{\overline{t-1}} = \texttt{get\_D}([\Phi^{\emptyset}_{\overline{t-1}},\ \Phi^{3\text{CRV}}_{\overline{t-1}} \times P^{3\text{CRV}}],\ \Phi^A)$$

   Beanstalk calculates $\Phi^{\emptyset^*}_{\overline{t-1}}$ from $\zeta^{\Phi}_{\overline{t-1}}$ as:

$$\Phi^{\emptyset^*}_{\overline{t-1}} = \frac{\zeta^{\Phi}_{\overline{t-1}}}{2}$$

   Beanstalk calculates $\Delta b^{\Phi}_{\overline{t-1}}$ for a given $\Phi^{\emptyset^*}_{\overline{t-1}}$, $\Phi^{\emptyset}_{\overline{t-1}}$ and B_{t-1} as:

$$\Delta b^{\Phi}_{\overline{t-1}} = \begin{cases} \max\left(\Phi^{\emptyset^*}_{\overline{t-1}} - \Phi^{\emptyset}_{\overline{t-1}}, -\frac{B_{t-1}}{100}\right) & \text{if } \Phi^{\emptyset^*}_{\overline{t-1}} - \Phi^{\emptyset}_{\overline{t-1}} < 0 \\ \min\left(\Phi^{\emptyset^*}_{\overline{t-1}} - \Phi^{\emptyset}_{\overline{t-1}}, \frac{B_{t-1}}{100}\right) & \text{else} \end{cases}$$

Therefore, we define $\Delta B_{\overline{t-1}}$ for a given $\Delta b^{\Phi}_{\overline{t-1}}$ as:

$$\Delta B_{\overline{t-1}} = \Delta b^{\Phi}_{\overline{t-1}}$$

14.7 Flood

The following liquidity pool is on the *Oracle Whitelist* and is therefore included in the calculation of ΔB_{t-1}:

14.7.1 Φ

1. **Pool Address:** The Φ liquidity pool address is 0xc9C32cd16Bf7eFB85Ff14e0c8603cc90F6F2eE49.

2. Δb_{t-1} **Calculation:** The shortage of Beans in the BEAN:3CRV Curve liquidity pool at the end of the previous *Season* (Δb^{Φ}_{t-1}) is calculated as the difference between the optimal number of Beans ($\Phi^{\not\emptyset^*}_{t-1}$) and the number of Beans ($\Phi^{\not\emptyset}_{t-1}$) in Φ at the end of the previous *Season*, such that $\Delta b^{\Phi}_{t-1},\ \Phi^{\not\emptyset^*}_{t-1},\ \Phi^{\not\emptyset}_{t-1} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$. $\Phi^{\not\emptyset^*}_{t-1}$ is calculated from $\Phi^{\not\emptyset}_{t-1}$, the number of 3CRV in Φ at the end of the previous *Season* ($\Phi^{3\text{CRV}}_{t-1}$), such that $\Phi^{3\text{CRV}}_{t-1} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, $P^{3\text{CRV}}$ and Φ^{A}.

   Beanstalk calculates a price invariant for Φ at the end of the previous *Season* (ζ^{Φ}_{t-1}), such that $\zeta^{\Phi}_{t-1} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, by calling the Curve `get_D` function on $\Phi^{\not\emptyset}_{t-1}$, $\Phi^{3\text{CRV}}_{t-1}$, $P^{3\text{CRV}}$ and Φ^{A} as:

$$\zeta^{\Phi}_{t-1} = \texttt{get\_D}([\Phi^{\not\emptyset}_{t-1},\ \Phi^{3\text{CRV}}_{t-1} \times P^{3\text{CRV}}],\ \Phi^{A})$$

   Beanstalk calculates $\Phi^{\not\emptyset^*}_{t-1}$ from ζ^{Φ}_{t-1} as:

$$\Phi^{\not\emptyset^*}_{t-1} = \frac{\zeta^{\Phi}_{t-1}}{2}$$

   Beanstalk calculates Δb^{Φ}_{t-1} for a given $\Phi^{\not\emptyset^*}_{t-1}$ and $\Phi^{\not\emptyset}_{t-1}$ as:

$$\Delta b^{\Phi}_{t-1} = \Phi^{\not\emptyset^*}_{t-1} - \Phi^{\not\emptyset}_{t-1}$$

Therefore, we define ΔB_{t-1} for a given Δb^{Φ}_{t-1} as:

$$\Delta B_{t-1} = \lceil \Delta b^{\Phi}_{t-1} \rceil$$

14.8 Market

Beanstalk supports the following exchanges on the *Market*:

14.8.1 Pods

Pods can be bought and sold in a decentralized fashion at the *Pod Market*.

14.8.1.1 Pod Orders

Anyone with Beans not in the *Silo* can *Order Pods*.

A *Pod Order* has four inputs:

1. The maximum number of *Pods* to be purchased;
2. The maximum place in the *Pod Line* (*i.e.*, the number of *Pods* that will become *Harvestable* before a given *Pod*) to purchase from;
3. The minimum number of Pods that can *Fill* the *Pod Order*; and
4. Either (a) a constant that represents the maximum price per *Pod* or (b) a piecewise polynomial function that determines the price per *Pod* by its current place in the *Pod Line*, denominated in Beans.

A *Pod Order* can be *Cancelled* at any time until it is entirely *Filled*. To facilitate instant clearance, Beans are locked in a *Pod Order* until it is entirely *Filled* or *Cancelled*. Beans can only be locked in a single *Pod Order* at a time.

14.8.1.2 Pod Listings

Pods that *Yield* from Beans that were *Sown* from a single call of the `sow` function form a *Plot*. Anyone with a *Plot* can *List* a whole or partial *Plot* to be sold for Beans.

A *Pod Listing* has six inputs:

1. The *Plot* being *Listed*;
2. The difference between the front of the portion of the *Plot* included in the *Pod Listing* from the front of the whole *Plot*, denominated in *Pods*, where a null input *Lists* from the back of the *Plot*;
3. The number of *Pods* in the *Plot* for sale, where a null input *Lists* the whole *Plot*;
4. The minimum number of Pods that can *Fill* the *Pod Listing*;
5. The maximum number of total *Harvestable Pods* over all *Seasons* before the *Pod Listing* expires; and
6. Either (a) a constant that represents the minimum price per *Pod* or (b) a piecewise polynomial function that determines the price per *Pod* by its current place in the *Pod Line*, denominated in Beans.

A *Pod Listing* can be *Cancelled* at any time until it is entirely *Filled*. *Plots* can only be *Listed* in a single *Pod Listing* at a time. *Pod Listings* are automatically *Cancelled* if the owner of the *Plot* transfers, or simultaneously includes in another *Listing*, any *Pods* in the *Plot*.

14.8.1.3 Clearance

An outstanding *Pod Order* can be entirely or partially *Filled* at any time by a *Pod* seller. If the *Pod Order* is partially *Filled*, the rest of the *Pod Order* remains *Ordered*. Similarly, an outstanding *Pod Listing* can be entirely or partially *Filled* at any time by a *Pod* buyer. If the *Pod Listing* is partially *Filled*, the rest of the *Pod Listing* remains *Listed*.

In instances where $0 < \Delta D_t$ causes a *Pod Order* and *Pod Listing* that previously were not overlapping to overlap, either the buyer or seller can *Fill* their *Order* or *Listing*, respectively, at their preferred price.

14.8.1.4 Future Work

The *Pod Market* is a work in progress. The following are potential improvements that can be implemented as one or more *BIPs*.

- Multiple *Plots* can be *Listed* in the same *Pod Listing*.
- Transferring or *Listing Pods* not *Listed* in a partial *Listing* should not *Cancel* the *Listing*.
- Overlapping *Pod Orders* and *Pod Listings* can be cleared automatically.
- *Deposited* Beans can be used to place *Pod Orders* once there is no *Withdrawal Freeze* upon *Withdrawal* from the *Silo*.

14.9 Depot

The following *Pipelines* to the *Depot* currently exist:

14.9.1 Curve

The Curve *Pipeline* allows anyone to call functions in any pool registered in any of the following Curve registries.

- 0xB9fC157394Af804a3578134A6585C0dc9cc990d4[41]
- 0x90E00ACe148ca3b23Ac1bC8C240C2a7Dd9c2d7f5[42]
- 0x8F942C20D02bEfc377D41445793068908E2250D0[43]

The following functions to interact with Curve pools can be called through the Curve *Pipeline*.

- `exchange(...)`
- `exchange_underlying(...)`
- `add_liquidity(...)`
- `remove_liquidity(...)`
- `remove_liquidity_imbalanced(...)`
- `remove_liquidity_one_token(...)`

14.9.2 Pipeline

The Pipeline *Pipeline* allows anyone to perform an arbitrary series of actions in the EVM in a single transaction by using 0xb1bE0000bFdcDDc92A8290202830C4Ef689dCeaa[44] as a sandbox for execution.

The following functions to interact with Pipeline can be called through the Pipeline *Pipeline*.

- `pipe(...)`
- `multiPipe(...)`
- `advancedPipe(...)`

41 etherscan.io/address/0xB9fC157394Af804a3578134A6585C0dc9cc990d4#readContract
42 etherscan.io/address/0x90E00ACe148ca3b23Ac1bC8C240C2a7Dd9c2d7f5#readContract
43 etherscan.io/address/0x8F942C20D02bEfc377D41445793068908E2250D0#readContract
44 etherscan.io/address/0xb1bE0000bFdcDDc92A8290202830C4Ef689dCeaa#readContract

14.10 Fundraisers

Fundraisers allow Beanstalk to issue *Pods* in exchange for assets pegged to V other than Beans, independent of the *Soil* minting schedule, in order to raise funds to facilitate payments in other currencies (*e.g.*, to cover the cost of an audit) without directly affecting Beanstalk's normal peg maintenance model. *Fundraisers* are created via *Beanstalk Improvement Proposals* and mint new Beans.

Each *Fundraiser* requires (1) the token address of the token to raise, (2) the number of tokens to raise (*i.e.*, the number of Beans to mint), and (3) the wallet address to send the tokens to upon completion of the *Fundraiser*.

Up to (2) assets pegged to V can be exchanged for 1 *Sown* Bean's *Yield* of *Pods* each, based on the *Temperature* at the time of the contribution to the *Fundraiser*. Tokens raised via a *Fundraiser* are automatically distributed to (3) upon completion of the *Fundraiser*.

The following *Fundraisers* have been approved via Beanstalk governance:

14.10.1 Trail of Bits Audit[45]

1. **Token Address:** The USDC token address is 0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48.
2. **Tokens to Raise:** The *Fundraiser* is for 347,440 USDC.
3. **Wallet to Send Tokens:** The tokens are sent to 0x925753106FCdB6D2f30C3db295328a0A1c5fD1D1 upon completion of the *Fundraiser*.

14.10.2 Omniscia Audit[46]

1. **Token Address:** The USDC token address is 0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48.
2. **Tokens to Raise:** The *Fundraiser* is for 140,000 USDC.
3. **Wallet to Send Tokens:** The tokens are sent to 0x925753106FCdB6D2f30C3db295328a0A1c5fD1D1 upon completion of the *Fundraiser*.

14.10.3 Omniscia Retainer[47]

1. **Token Address:** The USDC token address is 0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48.
2. **Tokens to Raise:** The *Fundraiser* is for 250,000 USDC.
3. **Wallet to Send Tokens:** The tokens are sent to 0x21DE18B6A8f78eDe6D16C50A167f6B222DC08DF7 upon completion of the *Fundraiser*.

[45] bean.money/bip-4
[46] bean.money/bip-5
[47] bean.money/bip-10

14.11 Glossary

The following conventions are used throughout this paper:

- Lower case Latin letters are unique values;
- Upper case Latin letters are totals or rates;
- Mathfrak style Latin letters are related to the Barn;
- Hebrew letters refer to assets used prior to the *Exploit* and recapitalized by the *Barn*;
- Subscripts are time, where t is the current *Season*, Ξ is the end of the current block, and $\ni$ is the current transaction; and
- Superscripts are modifiers.

The following variables and terms are used throughout this paper:

\$ - 1 US Dollar;

$\$^{\text{𝒃}(\urcorner)}$ - The USD price of 𝒃1 from the old BEAN:3CRV Curve pool;

$\$^{\text{𝒃}(\Phi)}$ - The USD price of 𝒃1 in the pool;

$\$^{\text{𝒃}(\Phi)}_{\Xi-1}$ - The flash-loan-resistant USD price of 𝒃1 from the BEAN:3CRV Curve pool;

$\$^{\text{LUSD}(\Omega))}$ - The USD price of 1 LUSD from the LUSD:3CRV Curve pool;

𝒃 - Bean;

$\text{𝒃}^{\min}$ - The minimum number of Beans received for *Converting Deposited* Φ within the *Silo*;

𝒃:y - A new liquidity pool that consists of Beans and y;

A^{BIP} - The total a^{BIP} for all passed *BIPs*;

A^q - The total awards for all committed *BIPs*

a^{BIP} - The award for submitting a *BIP* that gets accepted;

a^q - The award for successfully committing an approved *BIP* DEPRECATED;

a_t - The award for successfully calling the `sunrise` function for t;

$\mathfrak{A}$ - *Active Fertilizer*;

Active Fertilizer - The number of *Fertilizer* that have been bought but have not *Fertilized* all associated *Sprouts*;

AMM - Automated market maker;

Available Fertilizer - The number of *Fertilizer* that can be bought from Beanstalk in exchange for 1 USDC each;

B - The total Bean supply;

B^{BIP} - The total Beans minted via *BIP*;

Barn - The Beanstalk recapitalization facility;

BCM - The *Beanstalk Community Multisig*;

BDV - Bean-denominated-value;

Beanstalk Community Multisig - The owner of the Beanstalk contract;

Beanstalk Improvement Proposal - A Beanstalk governance proposal;

BIP - A *Beanstalk Improvement Proposal*;

Burnt - Sent to the null address;

$\beth$ - Old BEAN:ETH Uniswap V2 LP tokens;

$\beth$ - The current total number of $\beth$ in the current block;

$\beth^{\textcent}$ - The last traded price was a function of the current number of Beans in $\beth$;

$\beth^{\textcent}_{\bar{t}}$ - The time weighted average number of Beans in $\beth$ from the start of the current *Season* to the current block;

C_t - A *Stalkholder's* total *Seeds* during t;

$C_{\odot}$ - A *Stalkholder's Seeds* at the end of the block prior to the *Exploit*;

$C_{\otimes}$ - A *Stalkholder's Seeds* upon *Replant*;

c^{λ} - The number of *Seeds* per BDV of λ *Deposited*;

c^{λ}_t - The *Seeds* during t for a given *Deposit*;

Cancel - Revoke an offer to buy or sell;

Chop - Take an *Unripe* asset and burn through Beanstalk them to receive a portion of the associated *Ripe* asset;

Convert - Exchange one *Deposited* λ for another, within the *Silo*;

Convert Whitelist - The whitelist that permissions *Conversions* within the *Silo*;

D - The total number of *Unharvestable Pods*;

D_{γ} - The number of *Unharvestable Pods* that grew prior to the the *Farm* becoming *Oversaturated*;

d - The number of *Pods* that *Yield* from a given number of *Sown* $\textcent$;

$\mathfrak{D}$ - The total *Unfertilized Sprouts*;

$\mathfrak{d}$ - The number of *Sprouts* ultimately *Fertilized* by *Available Fertilizer* purchased with *Humidity* $\mathfrak{H}$;

DAO - Decentralized autonomous organization;

DeFi - Decentralized finance;

Deposit - An asset in the *Silo*;

Deposit Whitelist - The whitelist that permissions *Deposits* into the *Silo*;

Depositors - A wallet that has *Deposited* assets into the *Silo*;

Depot - Portion of the *Farm* that facilitates interactions with other Ethereum-native protocols through Beanstalk in a single transaction;

DEX - Decentralized exchange;

ΔB_{t-1} - The number of Beans that are minted and sold to return the price of $\textcent 1$ to its value peg;

$\Delta B_{\overline{t-1}}$ - The sum of liquidity and time weighted average shortages or excesss of Beans across $\emptyset$:y liquidity pools on the *Oracle Whitelist* over the previous *Season*;

Δb^{Φ}_{t-1} - The shortage of Beans in the BEAN:3CRV Curve liquidity pool at the end of the previous *Season*;

$\Delta b^{\Phi}_{\overline{t-1}}$ - The liquidity and time weighted average shortage or excess of Beans in the BEAN:3CRV Curve liquidity pool over the previous *Season*;

ΔD_t - The number of Pods that *Ripen* and become *Harvestable* at the beginning of each *Season*;

$\Delta\mathfrak{D}$ - The total *Sprouts Fertilized* by *Fertilizer*;

$\Delta\mathfrak{D}_t$ - The number of *Unfertilized Sprouts* that are *Fertilized* by *Active Fertilizer* and become *Rinsable* at the beginning of each *Season*;

ΔE^u_t - The difference in time it took for the Beans to be *Sown* in all but at most one *Soil* over the previous two *Seasons*;

$\Delta E^{u^{\text{first}}}_t$ - The time of the first *Sow* such that Beans are *Sown* in all but at most one *Soil* in each *Season*;

$\Delta\mathfrak{R}^{\emptyset}$ - The change in *Ripe* $\emptyset$for a given purchase of *Fertilizer*;

$\Delta\mathfrak{R}^{\Phi}$ - The change in *Ripe* Φ for a given $\Delta\mathfrak{S}_{\supset}$;

ΔS_t - The change in *Soil* from the beginning to the end of each *Season*;

$\Delta\mathfrak{S}_{\supset}$ - A purchase of *Fertilizer*;

$\Delta\mathfrak{X}$ - The change in $\mathfrak{X}$ between (1) φ or (2) the *Replant*, if $\varphi = 0$, and t;

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

$\urcorner$ - Old BEAN:3CRV Curve LP tokens;

$\urcorner^A$ - The A parameter of $\urcorner$;

$\urcorner_{\Xi-1}$ - The flash-loan-resistant total number of $\urcorner$;

$\urcorner^{\emptyset}_{\Xi-1}$ - The number of Beans in the old BEAN:3CRV Curve pool at the end of the last block;

$\urcorner^{3\text{CRV}}_{\Xi-1}$ - the number of 3CRV in the old BEAN:3CRV Curve pool at the end of the last block;

E - Ethereum block timestamps;

E_1 - The timestamp in the Ethereum block containing the Beanstalk deployment;

E_{BIP} - The end of a BIP's *Voting Period* DEPRECATED;

$E_{\sqsupset}$ - The timestamp of the last interaction with $\beth$;

E^*_{Ξ} - The current block timestamp;

E_t - The timestamp of the block containing the `sunrise` function that began t;

$E^{\min}_t$ - The minimum timestamp Beanstalk accepts a `sunrise` function call for a given t;

$E^{u^{\text{first}}}_t$ - The Ethereum timestamp of the first *Sow* in t such that there is at most one *Soil*;

E_q - The timestamp a BIP was committed DEPRECATED;

E_{Ψ} - The timestamp Beanstalk last *Unpaused*;

Earned $\emptyset$ - Beans paid to a *Stalkholder* after the last *Season* the *Stalkholder* called the `plant` function;

Enroot - Turn *Revitalized Stalk* and *Revitalized Seeds* into *Stalk* and *Seeds*;

ETH - Ether;

Exploit - The April 17th, 2022, governance exploit of Beanstalk.;

$f^{\lambda \to \lambda'}(z^{\lambda})$ - The function to determine the number of λ' received for *Converting* a given number of λ;

$f^{\lambda}(z^{\lambda})$ - The function to calculate the flash-loan-resistant Bean-denominated-value for a given number of λ *Deposited*;

$\mathfrak{F}$ - The total *Fertilizer*;

Farm - Where Bean peg maintenance and use of Beans take place;

Fertilized - Become redeemable;

Fertilizer - A limited debt issuance;

Field - The Beanstalk credit facility;

FIFO - First in, first out;

Fill - Match an outstanding offer to buy or sell;

Flood - When Beanstalk mints extra Beans and sells them directly in liquidity pools on the *Oracle Whitelist*;

Frozen - Unable to be *Claimed*;

Fundraisers - Allow Beanstalk to issue *Pods* in exchange for assets pegged to V other than Beans, independent of the *Soil* minting schedule;

$\zeta^{\urcorner}$ - The flash-loan-resistant price invariant for the BEAN:3CRV Curve pool;

ζ^{Φ} - The price invariant for the BEAN:3CRV Curve pool;

$\zeta^{\Phi}_{\Xi-1}$ - The flash-loan-resistant price invariant for the BEAN:3CRV Curve pool;

ζ^{Φ}_{t-1} - The price invariant for Φ at the end of the previous *Season*;

$\zeta^{\Phi}_{\overline{t-1}}$ - The liquidity and time weighted average price invariant for Φ over the previous *Season*;

ζ^{Ω} - The flash-loan-resistant price invariant for the LUSD:3CRV Curve pool;

G_t - A *Stalkholder's* total *Grown Stalk* that can be *Mown* during t;

g_t^{λ} - The *Grown Stalk* from *Seeds* that can be *Mown* during t to start earning Bean seigniorage for a given *Deposit* of a *Stalkholder* that last *Mowed* their *Grown Stalk* in $\varkappa$;

Grow - *Stalk* being created by *Seeds*;

Grown Stalk - *Stalk* that has been created by *Seeds* and not yet *Mown*;

η - The last *Season* a *Stalkholder* called the `plant` function;

$\eta^{\emptyset}$ - *Earned* $\emptyset$;

η^{c} - The *Plantable Seeds* associated with a *Stalkholder's* $\eta^{\emptyset}$ that can be *Planted* to start earning *Grown Stalk*;

$\eta^{c}_{\odot}$ - A *Stalkholder's Plantable Seeds* at the end of the block prior to the *Exploit*;

$\beth$ - Old BEAN:LUSD Curve LP tokens;

h_t - The *Temperature* during t;

h - The *Temperature*;

$\mathfrak{H}$ - The *Humidity*;

Harvest - Redeem;

Harvestable - Redeemable;

Harvested - Redeemed;

Humidity - The interest rate on *Fertilizer* purchases;

$K_{\odot}$ - A *Stalkholder's Stalk* at the end of the block prior to the *Exploit*;

$K_{\otimes}$ - A *Stalkholder's Stalk* upon *Replant*;

K_t - A *Stalkholder's* total *Stalk* during t;

$K^{\min}$ - The percentage of *Stalk* ownership necessary to submit a *BIP*;

k^{λ} - The number of *Stalk* per BDV of λ *Deposited*;

k_t^{λ} - The *Stalk* during t for a given *Deposit* of a *Stalkholder* that last *Mowed* their *Grown Stalk* in $\varkappa$;

$\varkappa$ - The last *Season* a *Stalkholder Mowed* their *Grown Stalk*;

L_i^{λ} - The total BDV of Z_i^{λ} when *Deposited*;

List - Create an offer to sell;

LP tokens - Liquidity pool tokens;

λ - $\textcircled{\scriptsize P}$ and other assets on the *Deposit Whitelist*;

Λ - The *Deposit Whitelist*;

M - The total Beans minted over all *Seasons*;

$\mathfrak{M}$ - The percentage of *Ripe* assets received for *Chopping* a pro-rata portion of *Unripe* assets;

Market - The Beanstalk-native DEX.;

Mow - Turn *Grown Stalk* into *Stalk*;

$N^{\textcircled{\scriptsize P}}$ - The total *Burnt* $\textcircled{\scriptsize P}$ over all *Seasons*;

Oracle Whitelist - The whitelist of pools included in the calculation of $\Delta B_{\overline{t-1}}$;

Order - Create an offer to buy;

Oversaturated - It is *Raining* and $R_{t-1}^D < R^{D^{\text{lower}}}$;

P^{3CRV} - The 3CRV virtual price;

$P^{\urcorner}$ - The $\urcorner$ virtual price;

$P^{\lrcorner}$ - The $\lrcorner$ virtual price;

$P_{\overline{t-1}}$ - The inferred liquidity and time weighted average price of $\textcircled{\scriptsize P}1$ compared to V over the previous *Season*;

P^{Φ} - The Φ virtual price;

P^{Ω} - The Ω virtual price;

$\mathfrak{P}^{\textcircled{\scriptsize P}}$ - The number of Beans received for *Chopping* a given $\mathfrak{z}^{\textcircled{\scriptsize P}}$;

$\mathfrak{P}^{\Phi}$ - The number of Φ received for *Chopping* a given $\mathfrak{z}^{\Phi}$;

Pause - Stop accepting `sunrise` function calls;

Paused - Beanstalk has stopped accepting `sunrise` function calls;

Pipeline - A connection between Beanstalk and another Ethereum-native protocol via the *Depot*;

Plant - Turn *Seeds* associated with *Earned* $\not\Phi$ into *Seeds* by *Depositing* the *Earned* $\not\Phi$ in the current *Season*;

Plantable Seeds - *Seeds* that can be *Planted*;

Plot - Beans *Sown* from a single call of the `sow` function;

Pod Line - The order of *Pods* that will become *Harvestable*;

Pod Listing - An offer to sell *Pods*;

Pod Order - An offer to buy *Pods*;

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

Pod Market - A Beanstalk-native DEX for *Pods*;

Pods - The primary debt asset of Beanstalk;

R^D - The *Pod Rate*;

R^D_{t-1} - The *Pod Rate* at the end of the previous *Season*;

$\mathfrak{R}^{\not\Phi}$ - *Ripe* $\not\Phi$;

$\mathfrak{R}^{\not\Phi}_{<\beth}$ - The *Ripe* $\not\Phi$ prior to a purchase of *Fertilizer*;

$\mathfrak{R}^{\Phi}$ - *Ripe* Φ;

Raining - $V < P_{\overline{t-1}}$;

Replant - Restart Beanstalk after the governance *Exploit*;

Revitalized Seeds - *Seeds* that have become *Enrootable*;

Revitalized Stalk - *Stalk* that have become *Enrootable*;

Rinsable - Redeemable;

Rinsable Sprouts - Redeemable *Sprouts*;

Rinse - Redeem;

Ripe assets - The assets received upon *Chopping Unripe* assets;

Ripen - Become *Harvestable*;

S - *Soil*;

S_t^{end} - The *Soil* supply at the end of that *Season*;

S_t^{min} - The *Minimum Soil* at the beginning of t;

S_t^{start} - The *Soil* supply at the beginning of each *Season*;

$\mathfrak{S}$ - The total *Fertilizer* sold;

$\mathfrak{S}_{<\beth}$ - The *Fertilizer* sold prior to a purchase of *Fertilizer*;

Season - Beanstalk-native discrete time;

Seed - A Beanstalk-native asset that *Grow* 1×10^{-4} *Stalk* each *Season* ;

Silo - The Beanstalk DAO;

Soil - An offer from Beanstalk to borrow Beans;

Sow - Lend Beans;

Sower - A Beanstalk creditor;

Sown - Lent;

Sprouts - Assets that can be redeemed for $\not\Beta 1$ if it has been *Fertilized*;

Stalk System - The Beanstalk-native mechanism for *Stalk*;

Stalk - The Beanstalk-native governance asset;

Stalkholder - A Beanstalk DAO member;

Sun - The Beanstalk-native execution and timekeeping mechanism;

Φ - BEAN:3CRV LP tokens;

$\Phi^{\not\Beta}$ - The number of Beans in the BEAN:3CRV Curve pool;

Φ^{3CRV} - The number of 3CRV in the BEAN:3CRV Curve pool;

Φ^{A} - The A parameter of Φ;

$\Phi^{\min}$ - The minimum number of Φ received for *Converting Deposited* Beans within the *Silo*;

$\Phi_{\Xi-1}$ - The flash-loan-resistant total number of Φ;

$\Phi_{\Xi-1}^{\not\Beta}$ - The number of Beans in the BEAN:3CRV Curve pool at the end of the last block;

$\Phi_{\Xi-1}^{\text{3CRV}}$ - The number of 3CRV in the BEAN:3CRV Curve pool at the end of the last block;

$\Phi_{t-1}^{\not\Beta}$ - The number of Beans in Φ at the end of the previous *Season*;

$\Phi_{t-1}^{\not\Beta^*}$ - The optimal number of Beans in Φ at the end of the previous *Season*;

Φ_{t-1}^{3CRV} - The number of 3CRV in Φ at the end of the previous *Season*;

$\Phi_{\overline{t-1}}^{\not\Beta}$ - The liquidity and time weighted average number of Beans in Φ over the previous *Season*;

$\Phi_{\overline{t-1}}^{\not\Beta^*}$ - The optimal liquidity and time weighted average number of Beans in Φ over the previous *Season*;

$\Phi_{\overline{t-1}}^{\text{3CRV}}$ - The time weighted average number of 3CRV in Φ over the previous *Season*;

φ - The *Season* a *Stalkholder* last called the `enroot` function;

φ_t^C - The number of *Revitalized Seeds* that can be *Enrooted* by a *Stalkholder* during t;

φ_t^K - The number of *Revitalized Stalk* that can be *Enrooted* by a *Stalkholder* during t;

t - The current *Season*;

Temperature - The interest rate on Bean loans;

Transfer - Send a *Deposited* asset;

U - The total *Sown* $\not\Beta$ over all *Seasons*;

u - The number of *Sown* $\not\Beta$;

$\mathfrak{U}$ - *Used Fertilizer*;

Unfertilized Sprouts - *Sprouts* not yet *Fertilized* by *Active Fertilizer*;

Unharvestable Pods - *Pods* that are not yet redeemable;

Unpause - Resume accepting `sunrise` function calls;

Unpaused - Beanstalk has resumed accepting `sunrise` function calls;

Unripe assets - Assets that can be *Chopped* to receive *Ripe* assets;

USD - 1 US Dollar;

Used Fertilizer - The number of *Fertilizer* that have been bought and *Fertilized* all associated *Sprouts*;

V - The value peg for $\emptyset$1;

$\mathfrak{V}$ - *Available Fertilizer*;

$\mathfrak{V}_{\mathfrak{H}}$ - *Available Fertilizer* purchased with *Humidity* $\mathfrak{H}$;

Voting Period - The period of time *Stalkholders* can vote on a *BIP*;

Withdraw - Remove from the *Silo*;

Withdrawal Freeze - The amount of time assets that have been removed from the *Silo* are *Frozen*;

Withdrawn assets - Assets that have been removed from the *Silo* and are still *Frozen*;

x - An existing ERC-20 Standard convertible stablecoin that (1) offers low-friction convertibility to V and (2) trades on an AMM against y;

x:y - An existing liquidity pool that consists of x and y;

$\mathfrak{X}$ - The percentage of *Fertilizer* sold;

$\mathfrak{X}_{\otimes}$ - The percentage of *Fertilizer* sold prior to *Replant*;

y - A liquid, decentralized network-native asset with endogenous value;

Yield - *Pods* being created from *Sown* $\emptyset$;

$\mathfrak{Z}^{\emptyset}$ - The total *Unripe* $\emptyset$;

$\mathfrak{Z}^{\Phi}$ - The current total *Unripe* Φ;

Z_i^{λ} - The total number of λ *Deposited* during *Season* i;

$\mathfrak{Z}_{\otimes}^{\Phi}$ - The total *Unripe* Φ at the time of *Replant*;

$\mathfrak{z}^{\emptyset}$ - *Unripe* $\emptyset$;

$\mathfrak{z}^{\emptyset^{\min}}$ - The minimum number of *Unripe* Beans received for *Converting Deposited Unripe* Φ within the *Silo*;

$\mathfrak{z}^{\Phi}$ - *Unripe* Φ;

$\mathfrak{z}^{\Phi^{\min}}$ - The minimum number of *Unripe* Φ received for *Converting Deposited Unripe* Beans within the *Silo*;

Ω - The LUSD:3CRV Curve pool;

Ω^{A} - The A parameter of Ω;

$\Omega_{\Xi-1}$ - The flash-loan-resistant total number of Ω;

$\Omega_{\Xi-1}^{\text{3CRV}}$ - The number of 3CRV in the LUSD:3CRV Curve pool at the end of the last block; and

$\Omega_{\Xi-1}^{\text{LUSD}}$ - The number of LUSD in the LUSD:3CRV Curve pool at the end of the last block.

14.12 Whitepaper Version History

The following is a complete version history of this whitepaper. Unless otherwise noted, references within this Whitepaper Version History are not updated to reflect later changes.

- 1.0.0 (August 6, 2021)
  - Original whitepaper.
- 1.0.1 (August 10, 2021) [Code Version 1.0.1 should have been 1.0.0.]
  - Updated Section 5 to reflect that the first *Season* began when the `init` function was called as part of the Beanstalk deployment.
  - Updated Section 6.4.3 to reflect that the first *Season* began when the `init` function was called as part of the Beanstalk deployment, and state that $P_{\bar{t}} = 1$ for each *Season* that contains a *Pause*.
  - Moved a paragraph from Section 6.4.3 to 6.4.4 for better flow.
  - Updated the definition of a^q in Section 6.4.5 to reflect the correct base commit award. [a^q was defined correctly in Version 1.0.0 but defined incorrectly in Versions 1.0.1 - 1.1.2.]
  - Updated Section 9.1 to reflect that the first *Season* began when the `init` function was called as part of the Beanstalk deployment.
- 1.1.0 (August 26, 2021)
  - Updated Section 6.3 to reflect the new *Stalk* equations, as amended by BIP-0.
  - Added t_f to the Glossary.
- 1.1.1 (September 15, 2021)
  - Added bean.money URL to the cover page.
- 1.1.2 (September 23, 2021)
  - Updated citation 16 with the correct URL for BIP-0.
- 1.1.3 (October 15, 2021) [Whitepaper Version 1.1.3 should have been 1.2.0. Code Version 1.1.2 should have been 1.2.0.]
  - Updated the definition of a^q in Section 6.4.5 to reflect the correct base commit award. [a^q was defined correctly in Version 1.0.0 but defined incorrectly in Versions 1.0.1 - 1.1.2.]
- 1.3.0 (November 11, 2021)
  - Updated Section 8.4.8 to reflect the latest *Weather* changes, as amended by BIP-2.[48]
  - Updated Section 11 to reflect an updated understanding of potential uses of Beanstalk.
  - Created an Appendix and moved Section 12 and Section 13 to the Appendix as Sections 12.1 and 12.2, respectively.
  - Updated Section 12.1 to reflect an updated understanding of potential uses of Beanstalk.
  - Added Section 12.3, a Whitepaper Version History, to the Appendix.

[48] bean.money/bip-2

- 1.3.1 (December 3, 2021)
  - Removed a sentence from the second paragraph of Section 6.2 to reflect the new *Stalk* equations, as amended by Pause Patch-0.
  - Updated Section 6.3 to reflect the new *Stalk* equations, as amended by Pause Patch-0.
  - Added a comma in the second paragraph of Section 8.3 for clarity.
  - Added f^{ϕ} to the Glossary.
  - Italicized *Stalk* in Whitepaper Version History changes for Version 1.1.0.
- 1.4.0 (December 10, 2021)
  - Modified the formatting of two equations and the language of the fifth paragraph in Section 6.3 for clarity.
  - Changed variables b_h, h and Λ_h to b_Ω, Ω and Λ_Ω, respectively, in Section 6.3 and the Glossary.
  - Updated Sections 7.1, 8, 8.1, 8.2, 8.3 and 8.4.5 to reflect the new *Soil* mechanism, as amended by BIP-6.[49]
  - Added h_t to the Glossary.
  - Corrected a typo in the change history for Whitepaper Version 1.3.1 in Section 12.3.
- 1.5.0 (December 18, 2021)
  - Modified the language of the seventh paragraph in Section 3 for clarity.
  - Switched all $>$ to $<$ for consistency and clarity.
  - Updated Sections 6.2, 6.3 and 9.6, and Figure 1, to reflect the new *Convert* mechanism, as amended by BIP-7.[50]
  - Updated Figure 2 to mirror the new design of Figure 1.
  - Modified the language of the second paragraph in Section 11 for consistency.
- 1.6.0 (January 12, 2022)
  - Modified the last sentence of the Abstract for better flow.
  - Changed a semicolon to a colon in the fourth paragraph of Section 1 for clarity.
  - Corrected a typo in the second paragraph of Section 3.
  - Updated the fourth and fifth paragraphs of Section 3 to reflect an updated understanding of potential uses of Beanstalk.
  - Modified the language of the fifth paragraph of Section 4 for consistency.
  - Modified the language of the first paragraph of Section 6.1 for clarity.
  - Updated the first paragraph of Section 6.2 to reflect the new *Withdrawal Freeze*, as amended by BIP-9.
  - Modified the language of the second paragraph of Section 6.2 for clarity.
  - Modified the language of the first, third and twelfth paragraphs of Section 6.3 for clarity.
  - Modified the equation for K_t for consistency.
  - Modified the language of the first paragraph of Section 6.4 for clarity.
  - Updated Section 6.4.1 to reflect the new governance policy, as amended by BIP-9.
  - Corrected a typo in the third paragraph of Section 6.4.2.

[49] bean.money/bip-6
[50] bean.money/bip-7

- Changed variable E_f to E_Ψ in Section 6.4.3 and the Glossary.
- Corrected typos in the first paragraph of Section 6.4.4 and the fourth paragraph of Section 6.4.5.
- Modified the language of the third paragraph of Section 6.4.5 for clarity.
- Modified the language of the first paragraph of Section 7 for consistency.
- Updated the second paragraph of Section 7.2 to reflect the new *Soil* policy, as amended by BIP-9.
- Updated Section 8.2 to reflect the new Bean supply policy, as amended by BIP-9.
- Corrected a typo and modified the language for clarity in the penultimate paragraph of Section 8.2.
- Modified the last equation in Section 8.2 for consistency.
- Updated Section 8.3 to reflect the new *Soil* supply policy, as amended by BIP-9.
- Modified the language of the second paragraph of Section 8.3 for clarity.
- Updated the equation for S_t^{start} in Section 8.3 to reflect the new *Soil* supply policy, as amended by BIP-9.
- Corrected a typo in the first paragraph of Section 8.4.1.
- Modified the language of the second paragraph of Section 8.4.3 for clarity.
- Modified the language of the first and second paragraphs of Section 8.4.4 for clarity.
- Updated Section 8.4.5 to reflect the new *Soil* supply policy, as amended by BIP-9.
- Modified the language of the first paragraph of Section 8.4.7 for clarity.
- Corrected a typo, and modified the language to reflect the new *Season of Plenty* timer, as amended by BIP-9, in the second paragraph of Section 8.2.
- Modified the language of the third paragraph of Section 9.1 for clarity.
- Modified the section titles of Sections 9.1, 9.2 and 9.4 for consistency.
- Modified the language of the second and third paragraphs of Section 9.4 for clarity.
- Modified the language of the first and second paragraphs of Section 9.6 for clarity.
- Modified the language of the third and fourth paragraphs of Section 9.6 to reflect current incentive structures.
- Corrected a typo in the second paragraph of Section 10.
- Modified the language of the fourth paragraph of Section 10 for accuracy.
- Updated the fifth paragraph of Section 11 to reflect an updated understanding of potential uses of Beanstalk.
- Modified the section title, and language of the first paragraph, of Section 12.1 to clarify the listed parameters are current.
- Modified the conventions in Section 12.2 to reflect consistency with regard to Latin letters only.
- Added $K^{\min}$, Λ^{Silo}, and ξ to the Glossary.
- Changed S_{t-1}^{end} to S_t^{end} in the Glossary for consistency.
- Removed B_t, $S_t^{\max}$, and $R_S^{\max}$ from the Glossary.
- Modified the language in the change histories for Versions 1.0.1, 1.1.0, 1.1.3, 1.3.1 in Section 12.3 for consistency.

- 1.7.0 (February 5, 2022)
  - Added a new Section 12.2 to describe the *Farmers Market* to the Appendix.
  - Changed *Deposit* in the Glossary for clarity.
  - Moved *Optimal State* in the Glossary to reflect correct alphabetical ordering.
  - Added *Cancel*, *Farmers Market*, *Fill*, *Listing*, *Plot*, *Pod Line*, *Pod Listing*, and *Withdrawal* to the Glossary.
- 1.8.0 (March 10, 2022)
  - Changed the fourth paragraph of Section 4 to reflect the update to the *Silo*, as amended by BIP-12.[51]
  - Changed Section 6 to reflect the update to the *Silo*, as amended by BIP-12.
  - Changed the third paragraph of Section 11 to reflect additional potential changes to the *Silo*.
  - Added c^{λ}, c_t^{λ}, $g^{\lambda}(z^{\lambda})$, k^{λ}, K_t^{λ}, l, λ, Λ, z^{λ} and Z_i^{λ} to the Glossary.
  - Changed G to μ, Λ to ϕ and Λ^{Silo} to ϕ^{Silo} in the Glossary.
  - Removed b_{Ω}, $c_t^{\not\Phi}$, c_t^{Λ}, $k_t^{\not\Phi}$, k_t^{Λ}, l_i^{Λ}, $\lambda^{\not\Phi}$, λ^{Λ}, Λ_{Ω}, $z_i^{\not\Phi}$, z_i^{Λ}, $z_i^{\Lambda:\not\Phi}$ and Ω from the Glossary.
- 1.9.0 (March 11, 2022)
  - Updated Figure 11 and Figure 12 to reflect the new *Weather* changes, as amended by BIP-13.[52]
  - Corrected a typo in the change history for Whitepaper Version 1.3.0 in Section 12.3.
- 1.9.1 (March 16, 2022)
  - Corrected Section 8.4.8 to reflect the new *Weather* changes, as amended by BIP-13.
  - Updated Whitepaper Version History links for Versions 1.6.0, 1.7.0, and 1.8.0.
- 1.9.2 (April 1, 2022)
  - Corrected a typo in the first paragraph of Section 6.2.
  - Updated the second paragraph of Section 6.2 to reflect the flash-loan-resistant nature of Bean-denominated-value.
  - Corrected the formatting of a^{BIP} and A^{BIP}.
  - Corrected Section 6.5.5 to reflect the correct rate and duration that a^q compounds.
  - Updated the equation for B in Section 8.4.5 to include B^{BIP}.
  - Corrected Section 8.4.8 to reflect the *Weather* changes when R^D equals $R^{D^{\text{lower}}}$, R^{D^*} or $R^{D^{\text{upper}}}$.
  - Added a new Section 12.2 to describe the *Silo Whitelist* to the Appendix.
  - Added a new Section 12.4 to describe *Fundraisers* to the Appendix.
  - Added B^{BIP}, BDV, $c^{\not\Phi}$, c^{ϕ}, c^{Φ}, E_{Ξ}, E_{ϕ}, E_{Ψ}, ζ^{Φ}, $g^{\not\Phi}(z^{\not\Phi})$, $g^{\phi}(z^{\phi})$, $g^{\Phi}(z^{\Phi})$, $g^{\text{3CRV}}(z^{\text{3CRV}})$, $k^{\not\Phi}$, k^{ϕ}, k^{Φ}, P^{Φ}, P^{3CRV}, Φ, $\phi_{\bar{t}^{\not\Phi}}$, ϕ_{Ξ}, $\phi_{\Xi}^{\not\Phi}$, $\Phi_{\Xi-1}$, $\Phi_{\Xi-1}^{\not\Phi}$ and $\Phi_{\Xi-1}^{\text{3CRV}}$ to the Glossary.
  - Corrected two typos in the Glossary.
- 1.9.3 (April 3, 2022)
  - Corrected typos in the eleventh paragraph of Section 8.4.5, Section 12.2, third paragraph of Section 12.4, Glossary and Whitepaper Version History changes for Version 1.1.0.

[51] bean.money/bip-12
[52] bean.money/bip-13

- 1.15.0 (April 7, 2022)
  - Moved to a new whitepaper versioning system such that Whitepaper Versions line up with *BIPs*.
  - Moved the definition of B to Section 8.4.1 for consistency and clarity.
  - Updated Section 8.4.5 to reflect the new method to measure demand for *Soil*, as amended by BIP-15.[53]
  - Added $\Delta E_t^{u^{\text{first}}}$ and $E_t^{u^{\text{first}}}$ to the Glossary.
  - Changed the definition of ΔE_t^u in the Glossary.
  - Removed $\Delta E_t^{u^{\text{last}}}$, ΔR_{t-1}^S, $\frac{\partial R^S}{\partial t}$, $\frac{\partial R^S}{\partial t}^{\text{upper}}$, $E_{t-1}^{u^{\text{first}}}$, R^S, $R^{S^{\text{min}}}$, $R_t^{S^{\text{end}}}$ and $R_t^{S^{\text{start}}}$ from the Glossary.
  - Added a comma for clarity in 18 instances in the Whitepaper Version History.
- 1.16.0 (April 11, 2022)
  - Modified Sections 12.2.2 and 12.2.3 for consistency.
  - Added Section 12.2.4 to reflect the addition of α to the *Silo Whitelist*, as amended by BIP-16.[54]
  - Added $\$^{\emptyset(\Phi)}$, $\$^{\text{LUSD}(\Omega))}$, α, c^α, ζ^Ω, $g^\alpha(z^\alpha)$, k^α, P^α, P^Ω, Φ^A, Ω, Ω^A, $\Omega_{\Xi-1}$, $\Omega_{\Xi-1}^{\text{LUSD}}$ and $\Omega_{\Xi-1}^{\text{3CRV}}$ to the Glossary.
  - Added the word "liquidity" for clarity in 6 instances in the Glossary.
  - Added the word "v2" for clarity in 3 instances in the Glossary.
  - Added the word "Curve" for clarity in 3 instances in the Glossary.
  - Moved the ordering of P^Φ and P^{3CRV} in the Glossary for consistency.
- 2.0.0 (August 6, 2022)
  - Completely overhauled the whitepaper to reflect the state of Beanstalk after the *Replant*.
- 2.0.1 (September 15, 2022)
  - Changed Section 14.6 to reflect the update to the the calculation of Δb_{t-1}^Φ, as amended by EBIP-2.[55]
  - Updated BIP links throughout the whitepaper to the Beanstalk Governance Proposals Github Repository.[56]
  - Corrected eight instances of improper punctuation in the Whitepaper Version History.
- 2.1.0 (October 5, 2022)
  - Changed the last paragraph in Section 5.4 to reflect the addition of $\lambda \rightarrow \lambda$ *Converts*, as amended by BIP-24.[57]
  - Corrected a typo in the last paragraph of Section 7.1 and the first paragraph of Section 7.3.1.
  - Added a new Section 14.4.1 to reflect the addition of $\lambda \rightarrow \lambda$ *Converts*, as amended by BIP-24.
  - Corrected the date of modification of Version 2.0.1 in the the Whitepaper Version History.

[53] bean.money/bip-15
[54] bean.money/bip-16
[55] bean.money/ebip-2
[56] github.com/BeanstalkFarms/Beanstalk-Governance-Proposals
[57] bean.money/bip-24

- 2.1.1 (October 6, 2022)
  - Corrected the BDV per token in Sections 14.5.2, 14.5.3 and 14.5.4.
- 2.2.0 (November 11, 2022)
  - Changed periods at the beginning of lists to colons for correctness.
  - Removed `()` when referring to functions throughout the whitepaper.
  - Changed Sections 14.8.1.1 and 14.8.1.2 to reflect the (1) addition of an input for *Pod Orders* and *Pod Listings* that specifies the minimum number of Pods that can be used to *Fill* them, as amended by EBIP-3 and (2) upgrade to the price per *Pod* input for *Pod Orders* and *Pod Listings* to support a piecewise polynomial function that determines the price per *Pod* by its current place in the *Pod Line*, denominated in Beans, as amended by BIP-29.[58,59]
  - Corrected a typo in the first paragraph of Section 14.8.1.3.
  - Removed the future work item regarding support for arbitrary pricing functions in *Pod Orders* and *Pod Listings* in Section 14.8.1.4, as amended by BIP-29.
  - Updated intro to the Glossary for clarity.
  - Added $K^{\min}$ to the Glossary.
  - Moved Φ in the Glossary.
  - Corrected a typo in the definition of $\Delta\mathfrak{R}^{\emptyset}$ in the Glossary.
  - Removed duplicate definitions of *Rinsable* and *Unfertilized Sprouts* from the Glossary.
  - Corrected a typo in the Whitepaper Version History intro.
- 2.3.0 (December 8, 2022)
  - Updated the award for successfully calling the `sunrise` in Section 4, as amended by BIP-30.[60]
  - Changed Gnosis to Safe in Section 5.5.5.
  - Updated citation 29 with the new link for the *Beanstalk Community Multisig*.
  - Updated Section 14.9 to reflect the addition of Pipeline *Pipeline* to *Depot*, as amended by BIP-30.
  - Updated BIP links throughout the whitepaper to bean.money links that redirect to an Arweave upload of the given BIP.
  - Updated Snapshot proposal links throughout the whitepaper to bean.money links that redirect to an Arweave upload of the given proposal.
  - Capitalized Whitepaper Version throughout the Whitepaper Version History for consistency.
  - Updated Whitepaper Version links in the Whitepaper Version History to the Beanstalk Whitepaper GitHub Repository.[61]
  - Added citations for EBIP-3 and BIP-29 in the Whitepaper Version History.
  - Corrected a citation formatting error under Version 2.0.1 in the Whitepaper Version History.
  - Corrected a typo under Version 1.16.0 in the Whitepaper Version History.

[58] bean.money/ebip-3
[59] bean.money/bip-29
[60] bean.money/bip-30
[61] github.com/BeanstalkFarms/Beanstalk-Whitepaper