Beanstalk: A Permissionless Fiat Stablecoin Protocol

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-native, fiat stablecoin protocol that issues an ERC-20 Standard 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 ................................................. 5
  1.1 Convertible Stablecoins ................................. 5
  1.2 Non-convertible Stablecoins ......................... 6
  1.3 Beanstalk .............................................. 7

2 Previous Work ............................................. 7

3 Farm .......................................................... 8

4 Sun ............................................................ 8

5 Silo ............................................................ 10
  5.1 The Stalk System ......................................... 10
  5.2 Deposit Whitelist ........................................ 10
  5.3 Deposits, Withdrawals, Transfers and Conversions ....... 11
  5.4 Calculating Stalk and Seeds ......................... 11
  5.5 Governance ............................................... 12
    5.5.1 Participation ....................................... 13
    5.5.2 Voting Period ...................................... 13
    5.5.3 Pause ............................................... 13
    5.5.4 Beanstalk Improvement Proposals ............... 14
    5.5.5 Beanstalk Community Multisig .................... 14

6 Field .......................................................... 14
  6.1 Soil ..................................................... 14
  6.2 Pods ..................................................... 14
  6.3 Temperature ............................................ 15

7 Barn .......................................................... 15
  7.1 Fertilizer ............................................... 16
  7.2 Humidity ................................................. 16
  7.3 Recapitalization ....................................... 16
    7.3.1 Available Fertilizer ............................ 17
    7.3.2 Revitalized Stalk and Seeds ................... 17
    7.3.3 Unripe Assets ..................................... 18
    7.3.4 Ripe Assets ........................................ 18
    7.3.5 Chopping ........................................... 18

8 Peg Maintenance ............................................ 20
  8.1 Ideal Equilibrium ...................................... 20
  8.2 Decentralized Price Oracle .......................... 20
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\textsuperscript{10} (DAI) and Abracadabra\textsuperscript{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\textsuperscript{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.\textsuperscript{13} However, despite this rapid increase in supply, the borrowing rates on exogenous value convertible USD stablecoins have historically been higher\textsuperscript{14} than borrowing rates on USD.\textsuperscript{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\textsuperscript{16}) have failed. While hybrid value convertible stablecoins (e.g., FRAX\textsuperscript{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\textsuperscript{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.

\textsuperscript{10} makerdao.com  
\textsuperscript{11} abracadabra.money  
\textsuperscript{12} liquidity.org  
\textsuperscript{13} stablecoinindex.com/marketcap  
\textsuperscript{14} app.aave.com/markets  
\textsuperscript{15} newyorkfed.org/markets/reference-rates/effr  
\textsuperscript{16} allcryptowhitepapers.com/terra-whitepaper  
\textsuperscript{17} frax.finance  
\textsuperscript{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 compositability 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 $\$1$ 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 $\text{gm}$ 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 $\mathcal{D}$ and other assets ($\lambda$) on the Deposit Whitelist ($\Lambda$) (i.e., $\mathcal{D} \subset \lambda \in \Lambda$) for participation in governance of Beanstalk upgrades and passive contribution to security, stability and liquidity. Anyone can become a Stalkholder by Depositing $\lambda$ 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 $\text{gm}$ function call is committed. When Beanstalk accepts the $\text{gm}$ function call, the necessary code is executed.

Beanstalk only accepts one $\text{gm}$ function call per Season. Beanstalk accepts the first $\text{gm}$ 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 `gm` function call for a given \( t \) \((E^t_{\text{min}})\), \( \forall E^t_{\text{min}} \) such that \( 1 < t \), and \( E_1 \) is:

\[
E^t_{\text{min}} = 3600 \left( \left\lfloor \frac{E_1}{3600} \right\rfloor + t \right)
\]

The cost to execute the `gm` 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 `gm` function call with newly minted Beans.

To encourage regular `gm` function calls even during periods of congestion on the Ethereum network while minimizing cost, the award for successfully calling the `gm` function for \( t \) \((a_t)\) is based on (1) an approximation of the cost to call the `gm` function in Beans in the current block \((\varnothing_\Xi)\), (2) the inter-block maximum extractable value (MEV) manipulation resistant time weighted average (TWA) Bean reserves in the Multi Flow Pump\(^{22,23}\) of the BEAN:ETH Well\(^{24,25}\) from the beginning of the Season to the current transaction \((\Theta^{\text{SMA}}_{\text{ETH},0,t})\), and (3) the minimum number of Beans that must be in the BEAN:ETH Well in order for the oracle to be considered \((\Theta^\text{min}(\text{S})\)) such that \( a_t, \varnothing_\Xi, \Theta^{\text{SMA}}_{\text{ETH},0,t}, \Theta^\text{min}(\text{S}) \in \left\{ j \times 10^{-6} \mid j \in \mathbb{Z}^+ \right\} \), and compounds 1% every additional second that elapses past \( E^t_{\text{min}} \) for 300 seconds.

\(\varnothing_\Xi\) is based on approximations of (1) the gas used to execute the `gm` function call \((\varnothing)\), (2) the gas fee of the current block denominated in Wei \((\varpi_\Xi)\), such that \( \varnothing, \varpi_\Xi \in \mathbb{Z}^+ \), and (3) the current price of ETH in Beans \((\vartheta)\), such that \( \vartheta \in \{ j \times 10^{-6} \mid j \in \mathbb{Z}^+ \} \), up to a maximum of 100 Beans.

Beanstalk calculates \( \varnothing \) as the difference between `gasmleft`\(^{26}\) at the beginning and end of the `gm` function call \((\varsigma)\), such that \( \varsigma \in \mathbb{Z}^+ \), up to a maximum of \( 5 \times 10^5 \) gas:

\[
\varnothing = \min(\varsigma + 10^5, 5 \times 10^5)
\]

We define \( \varpi_\Xi \) as the result of `block.basefee`\(^{27}\) read through a separate contract\(^{28}\) plus a 5 Wei buffer to account for the priority fee:

\[
\varpi_\Xi = \text{block.basefee} + 5
\]

\( \vartheta \) is based on \( \Theta^{\text{SMA}}_{\text{ETH},0,t} \) and the inter-block MEV manipulation resistant TWA ETH reserves in the Multi Flow Pump of the BEAN:ETH Well from the beginning of the Season to the current transaction \((\Theta^{\text{SMA}}_{\text{ETH},0,t})\), such that \( \Theta^{\text{SMA}}_{\text{ETH},0,t} \in \left\{ j \times 10^{-18} \mid j \in \mathbb{Z}^+ \right\} \):

\[
\vartheta = \frac{\Theta^{\text{SMA}}_{\text{ETH},0,t} \times 10^{18}}{\Theta^{\text{SMA}}_{\text{ETH},0,t}}
\]

Therefore, we define \( \varnothing_\Xi \) for a given \( \varnothing, \varpi_\Xi \) and \( \vartheta \) as:

\[
\varnothing_\Xi = \min(\varnothing \times \varpi_\Xi \times \vartheta + 3, 100)
\]

\(^{22}\) basin.exchange/multi-flow-pump.pdf
\(^{23}\) etherscan.io/address/0xBA510f10E3095B83a0F33aa9ad2544E22570a87C
\(^{24}\) basin.exchange/basin.pdf
\(^{25}\) Any italicized terms not defined herein are defined by Basin.
\(^{26}\) docs.soliditylang.org/en/v0.7.6/units-and-global-variables.html#block-and-transaction-properties
\(^{27}\) docs.soliditylang.org/en/v0.8.7/units-and-global-variables.html#block-and-transaction-properties
\(^{28}\) etherscan.io/address/0x84292919cB64b590C0131550483707E43Ef223aC#code
Therefore, we define $a_t$ for a given $\Theta_S, \Theta_{\delta_0, \Theta}, \Theta_{\min(\lambda)}$, the timestamp of the current block ($E_\Xi$), and $E_{t_{\min}}$ as:

$$a_t = \begin{cases} 100 & \text{if } t = 1 \mid \Theta_S < \Theta_{\min(\lambda)} \\
\frac{\Theta_S}{\Xi} \times 1.01^{\min(E_\Xi - E_{t_{\min}}, 300)} & \text{else} \end{cases}$$

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, $1 \times 10^{-4}$ additional Stalk Grows from each Seed. Grown Stalk become Stalk when Mown. Grown Stalk from $\lambda$ Deposits are automatically Mown each time a Stalkholder interacts with $\lambda$ in the Silo (i.e., Deposit, Withdraw, Convert, Transfer, Plant and Enroot), or when they call the mow or mowMultiple function with $\lambda$.

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 $\Lambda$ via Beanstalk governance. $\hat{\lambda}$ is always on the Deposit Whitelist.

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

\( \lambda \) can be Deposited into, Withdrawn from and Converted within, the Silo at any time.

Beanstalk rewards Stalk and Seeds to Depositors immediately upon Depositing \( \lambda \) into the Silo based on its BDV when Deposited, \( k^\lambda \) and \( c^\lambda \). Deposits implement the ERC-1155 Standard.\(^{29}\)

Upon a Deposited asset’s Withdrawal from the Silo, the Deposit itself is burned and the number of Stalk, Seeds, and Stalk from Seeds rewarded to it must also be forfeited.

The number of Stalk, Seeds, and Stalk from Seeds rewarded to a Deposit are included in its Transfer to another address.

Conversions of Deposited \( \lambda \) to Deposited \( \lambda' \) 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 \( \lambda' \) received for Converting a given number of \( \lambda \) \((f^\lambda \to \lambda'(z^\lambda))\), such that \( f^\lambda \to \lambda'(z^\lambda) : \{j \times 10^{-\lambda} | j \in \mathbb{Z}^+\} \to \{j \times 10^{-\lambda'} | j \in \mathbb{Z}^+\} \), where \( z^\lambda \) is the number of \( \lambda \) Converted.

\[ \]  

Figure 1: Silo

5.4 Calculating Stalk and Seeds

A Stalkholder’s total Stalk is the sum of the Stalk for each of their Deposits and Earned \( \$ \) (\( \eta^\$ \)), such that \( \eta^\$ \in \{j \times 10^{-6} | j \in \mathbb{N}\} \). Earned \( \$ \) are Beans paid to a Stalkholder after the last time the Stalkholder called the plant function (\( \eta \)).\(^{30,31,32}\) Beans minted to the Silo are distributed to Stalkholders and become Earned \( \$ \) 10 blocks past the beginning of the Season in which they were minted. Earned \( \$ \) automatically earn Stalk. The next time the Stalkholder calls the plant function, Earned \( \$ \) are Deposited and the associated Seeds are Planted to start Growing Stalk.

When a Stalkholder Deposits \( \lambda \), they update the total number of \( \lambda \) Deposited during Season \( i \) (\( Z_i^\lambda \)) and its total BDV when Deposited (\( L_i^\lambda \)), such that \( Z_i^\lambda, L_i^\lambda \in \{j \times 10^{-6} | j \in \mathbb{Z}^+\} \), as \( Z_i^\lambda := z^\lambda \) and \( L_i^\lambda := f^\lambda(z^\lambda) \). Beanstalk stores a map of each Stalkholder’s Deposits that are still in the Silo, from Stalkholder to Deposit ID to Deposit totals (i.e., \( (Z_i^\lambda, L_i^\lambda) \)). Deposit ID is the concatenation of the \( \lambda \) token address and the maximum Grown Stalk per BDV of \( \lambda \) at the time of Deposit.

The Stalk for a given Deposit are determined by its duration of Deposit, BDV when Deposited, \( k^\lambda, c^\lambda \) and the last Season the Stalkholder Mowed their Grown Stalk from \( \lambda \) Deposits (\( x^\lambda \)).

\(^{29}\) ethereum.org/en/developers/docs/standards/tokens/erc-1155  
\(^{30}\) bean.money/bip-0  
\(^{31}\) bean.money/bip-0  
\(^{32}\) bean.money/bip-21
The Stalk during $t$ for a given Deposit of a Stalkholder that last Mowed their Grown Stalk from $\lambda$ Deposits in $v^\lambda (k^\lambda_t)$, such that $k^\lambda_t \in \{ j \times 10^{-10} \mid j \in \mathbb{Z}^+ \}$, is:

$$k^\lambda_t = L^\lambda_i \left( k^\lambda + \frac{c^\lambda(t - x^\lambda)}{10000} \right)$$

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

$$K_t = \sum_{\lambda \in \Lambda} \sum_{i=1}^{x^\lambda} k^\lambda_i + \eta^\delta$$

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

$$g^\lambda_t = L^\lambda_i \left( \frac{c^\lambda(t - x^\lambda)}{10000} \right)$$

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:

$$G_t = \sum_{\lambda \in \Lambda} \sum_{i=1}^{x^\lambda} g^\lambda_i$$

The Seeds for a given Deposit are determined by its BDV when Deposited and $c^\lambda$.

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

$$c^\lambda_t = c^\lambda L^\lambda_i$$

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

$$C_t = \sum_{\lambda \in \Lambda} \sum_{i=1}^{x^\lambda} c^\lambda_i$$

The Plantable Seeds associated with a Stalkholder’s $\eta^\delta$ that can be Planted to start earning Grown Stalk ($\eta^\delta_c$), such that $\eta^\delta_c \in \{ j \times 10^{-6} \mid j \in \mathbb{N} \}$, is:

$$\eta^\delta_c = c^\delta \times \eta^\delta$$

When a Stalkholder Withdraws $\lambda$, 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 $\lambda$, 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 Grown Stalk per BDV to retain its Grown Stalk from Seeds, and BDV if it is higher. When Converting multiple $\lambda$ Deposits, their Grown Stalk per BDV amounts 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 $\lambda$ owner can become a Stalkholder and participate in Beanstalk governance by Depositing $\lambda$ into the Silo to earn Stalk.

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

The submitter of a BIP must own more than $K_{\text{end}}$, such that $K_{\text{end}} \in \{ j \times 10^{-10} \mid j \in \mathbb{N}, j \leq 10^{10} \}$, percent of total outstanding Stalk at the end of the Voting Period in order for a BIP to be able to pass.

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

Beanstalk only accepts votes in favor of BIPs. A Stalkholder’s vote is counted in proportion to their Stalk at the beginning of the Voting Period that still exists. Stalkholders have the ability to delegate their vote to any other user.

5.5.2 Voting Period

A Voting Period opens when the Snapshot proposal for a BIP can be voted on and ends approximately 168 Seasons later, 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 beginning of the Voting Period that still exists is voting in favor of the BIP, it fails; and
- More than half of the total outstanding Stalk at the beginning of the Voting Period that still exists is voting in favor of the BIP, it passes.

If at any time 24 hours or more after the beginning and before the end of the Voting Period more than two-thirds of the total outstanding Stalk is voting 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 $gm$ function call. When Unpaused, the $gm$ function can be called at the beginning of the next hour.

For a given timestamp of last Unpause ($E_\Psi$) during Season $t'$, we define $E_{t'}^{\text{min}} \lor E_t^{\text{min}}$ such that $t' < t$ as:

$$E_{t'}^{\text{min}} = 3600 \left( \left\lceil \frac{E_\Psi}{3600} \right\rceil + t - t' \right)$$
5.5.4 Beanstalk Improvement Proposals

Beanstalk implements EIP-2535.\textsuperscript{35} Beanstalk is a diamond with multiple facets. Beanstalk supports multiple simultaneous BIPs with independent Voting Periods.

A BIP has three inputs: (1) a list of facets and functions to add and remove upon commit, (2) a function to run upon commit and (3) the Ethereum address of the contract with (2).

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\textsuperscript{36} 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 remove governance entirely, revoking 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. The Morning is the first $Q$, such that $Q \in \mathbb{Z}^+$, blocks of each Season. Beanstalk changes the Soil and Temperature at the beginning of each block of the Morning according to the peg maintenance mechanism.

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. $\varnothing 1$ is Sown in one Soil. Beanstalk permanently removes Sown $\varnothing$ from the Bean supply.

When Beanstalk is willing to borrow more Beans to remove them from the Bean supply, it creates more Soil. Beanstalk changes the Minimum Soil ($S_{min}^{q}$) in block $q$, such that $q \in \mathbb{Z}^+$, $q \leq Q$, of $t$, such that $S_{min}^{q} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, according to the peg maintenance mechanism. During the Morning of each Season, the Minimum Soil is the result of a Dutch auction.

6.2 Pods

Pods are the primary debt asset of Beanstalk. Beanstalk never defaults on debt: Pods automatically Yield from Sown $\varnothing$ and never expire.

\textsuperscript{35} eips.ethereum.org/EIPS/eip-2535
\textsuperscript{36} app.safe.global/eth:0xa9bA2C40b263843C04d344727b954A545c81D043
In the future, when the average price of $1$ is above its value peg over a Season, Pods Ripen and become Harvestable (redeemable) for $1$ at anytime. Pods Ripen on a first in, first out (FIFO) basis: 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 $\#1$ 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 $\#1$.

The number of Pods ($d$) that Yield from a given number of Sown $\#1$ ($u$), such that $d, u \in \{j \times 10^{-6} | j \in \mathbb{Z}^+\}$, Sown with a given $h$ is:

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

Beanstalk changes the Maximum Temperature it is willing to offer each Season ($h_{t_{\text{max}}}$), such that $h_{t_{\text{max}}} \in \mathbb{Z}^+$, at the beginning of each Season according to the peg maintenance mechanism. During the Morning of each Season, the Temperature is the result of a Dutch auction.\footnote{en.wikipedia.org/wiki/Dutch_auction}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{field.png}
\caption{Field}
\end{figure}

7 Barn

The Barn is the Beanstalk recapitalization facility, being used for the Beanstalk Replant.\footnote{bean.money/bfp-72} \footnote{bean.money/barn}

\footnote{en.wikipedia.org/wiki/Dutch_auction}
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 (\(V\)) as the number of Fertilizer that can be bought from Beanstalk in exchange for 1 USDC each, Active Fertilizer (\(A\)) as the number of Fertilizer that have been bought but have not Fertilized all associated Sprouts, and Used Fertilizer (\(U\)), such that \(V, A, U \in \mathbb{N}\), as the number of Fertilizer that have been bought and Fertilized all associated Sprouts.

In the future, when the average price of \(\phi\)1 is above its value peg over a Season, Active Fertilizer Fertilizes Sprouts such that they become Rinsable (redeemable) for \(\phi\)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.

7.2 Humidity

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

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

\[
d = V_H \times \left(1 + \frac{H}{100}\right)
\]

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., \(\phi\) and BEAN:3CRV Curve LP tokens (\(\Phi\)), 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 (\(\mathfrak{G}\)), such that \(\mathfrak{F}, \mathfrak{G} \in \mathbb{Z}^+\). \(\mathfrak{F}\) is a function of the current total Unripe \(\Phi (3^\Phi)\) and the total Unripe \(\Phi\) at the time of Replant (\(3^\Phi\)), such that \(3^\Phi, 3^\Phi \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}\). \(\mathfrak{G}\) is the sum of Active Fertilizer and Used Fertilizer.

We define \(\mathfrak{F}\) for a given \(3^\Phi\) and \(3^\Phi\) as:

\[
\mathfrak{F} = \frac{7.7 \times 10^7 \times 3^\Phi}{3^\Phi}
\]

We define \(\mathfrak{G}\) for a given \(\mathfrak{A}\) and \(\mathfrak{U}\) as:

\[
\mathfrak{G} = \mathfrak{A} + \mathfrak{U}
\]

Therefore, we define \(\mathfrak{V}\) for a given \(\mathfrak{F}\) and \(\mathfrak{G}\) as:

\[
\mathfrak{V} = \mathfrak{F} - \mathfrak{G}
\]

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 (\(x_\square\)), such that \(x_\square \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}\). As the percentage of Fertilizer sold (\(x\)), such that \(x \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}\), 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 \(x\) for a given \(\mathfrak{G}\) and \(\mathfrak{F}\) as:

\[
x = \frac{\mathfrak{G}}{\mathfrak{F}}
\]

A Stalkholder’s Stalk upon Replant (\(K_\square\)) given \(x_\square\) and their Stalk at the end of the block prior to the Exploit (\(K_\bigcirc\)), such that \(K_\square, K_\bigcirc \in \{j \times 10^{-10} \mid j \in \mathbb{Z}^+\}\), is:

\[
K_\square = x_\square \times K_\bigcirc
\]

A Stalkholder’s Seeds upon Replant (\(C_\square\)) given \(x_\square\), their Seeds at the end of the block prior to the Exploit (\(C_\bigcirc\)) and their Plantable Seeds at the end of the block prior to the Exploit (\(n_\square\)), such that \(C_\square, C_\bigcirc, n_\bigcirc \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}\), is:

\[
C_\square = x_\square \times (C_\bigcirc + n_\bigcirc)
\]

The number of Revitalized Stalk (\(\varphi_\square\)), such that \(\varphi_\square \in \{j \times 10^{-10} \mid j \in \mathbb{Z}^+\}\), and Revitalized Seeds (\(\varphi_\bigcirc\)), such that \(\varphi_\bigcirc \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}\), that can be Enrooted by a Stalkholder during \(t\) are functions of the change in \(x\) (\(\Delta x\)), such that \(\Delta x \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}\), between (1) the Season they last called the enroot function (\(\varphi\)) or (2) the Replant if they have never Enrooted their Revitalized Stalk and Revitalized Seeds (i.e., \(\varphi = 0\)), and \(t\), and \(K_\bigcirc\) or \(C_\bigcirc\) and \(n_\bigcirc\), respectively.

We define \(\Delta x\) for a given Stalkholder that last Enrooted their Revitalized Stalk and Revitalized Seeds in \(\varphi\) as:

\[
\Delta x = \begin{cases} 
  x_t - x_\varphi & \text{if } \varphi > 0 \\
  x_t - x_\square & \text{else}
\end{cases}
\]
We define $\phi^K_t$ for a given $\Delta X$ and $K_\circ$ as:

$$\phi^K_t = \Delta X \times K_\circ$$

We define $\phi^C_t$ for a given $\Delta X$ and $C_\circ$ as:

$$\phi^C_t = \Delta X \times (C_\circ + \eta_\circ)$$

### 7.3.3 Unripe Assets

Holders of Beans at the end of the block prior to the *Exploit* received *Unripe* $\delta$ ($3^\delta$), such that $3^\delta \in \{ j \times 10^{-6} | j \in \mathbb{Z}^+ \}$, at a 1:1 ratio. Holders of $\lambda$ not *Deposited* at the end of the block prior to the *Exploit* received *Unripe* $\Phi$ ($3^{\Phi}$), such that $3^{\Phi} \in \{ j \times 10^{-6} | j \in \mathbb{Z}^+ \}$, at a ratio of 1 $3^{\Phi}$ per BDV of $\lambda$ held at the end of the block prior to the *Exploit*. Holders of $\lambda$ *Deposited* at the end of the block prior to the *Exploit* received $3^\Phi$ at a ratio of 1 $3^\Phi$ per the maximum of the BDV of $\lambda$ *Deposits* at the end of the block prior to the *Exploit* and at the time of *Deposit*, per *Deposit*.

### 7.3.4 Ripe Assets

The number of *Ripe* assets (i.e., *Ripe* $\delta$ ($3^\delta$), such that $3^\delta \in \{ j \times 10^{-6} | j \in \mathbb{Z}^+ \}$, and *Ripe* $\Phi$ ($3^{\Phi}$), such that $3^{\Phi} \in \{ j \times 10^{-18} | j \in \mathbb{Z}^+ \}$), increases as more *Fertilizer* is sold.

The change in *Ripe* $\delta$ ($\Delta 3^\delta$) for a given purchase of *Fertilizer* ($\Delta S_\lambda$) is a function of the total *Unripe* $\delta$ ($3^\delta$), the *Ripe* prior to the purchase ($\delta_{<\circ}$), such that $\delta_{<\circ} \in \mathbb{Z}^+$, and the *Fertilizer* sold prior to the purchase ($S_{<\circ}$), such that $S_{<\circ} \in \mathbb{Z}^+$. We define $\Delta 3^\delta$ for a given $\Delta S_\lambda$, $3^\delta$, $\delta_{<\circ}$ as:

$$\Delta 3^\delta = \frac{\Delta S_\lambda \times (3^\delta - \delta_{<\circ})}{3^\delta - \delta_{<\circ}}$$

The change in *Ripe* $\Phi$ ($\Delta 3^{\Phi}$), such that $\Delta 3^{\Phi} \in \{ j \times 10^{-18} | j \in \mathbb{Z}^+ \}$, is the result of calling the `calc_token_amount` function on the Curve Zap contract for a given $\Delta \delta_{<\circ}$.

We define $\Delta 3^{\Phi}$ for a given $\Delta \delta_{<\circ}$ as:

$$\Delta 3^{\Phi} = \text{calc_token_amount}(\Phi, [0.866616 \times \Delta \delta_{<\circ}, 0, \Delta \delta_{<\circ}, 0], \text{true})$$

### 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* ($\Delta \delta$) and the total *Unfertilized Sprouts* (i.e., *Sprouts* not yet *Fertilized* by *Active Fertilizer*) ($\delta$), such that $\mathfrak{M}$, $\Delta \delta$, $\delta \in \{ j \times 10^{-6} | j \in \mathbb{Z}^+ \}$.

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

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

The number of Beans received for *Chopping* a given $3^\delta$ ($\mathfrak{M}^3$), such that $\mathfrak{M}^3 \in \{ j \times 10^{-6} | j \in \mathbb{Z}^+ \}$, is a function of $\mathfrak{M}$, $\mathfrak{M}^3$ and $3^\delta$. 

---

40 bean.money/bip-20
41 etherscan.io/address/0xA79828DF1850E8a3A3064576f380D90aECDD3359#code
We define $\mathcal{P}^\phi$ for a given $\phi$, $\mathcal{M}$ and $\mathcal{R}$ as:

$$\mathcal{P}^\phi = \frac{\phi \times \mathcal{M} \times \mathcal{R}}{3^\phi}$$

The number of $\Phi$ received for Chopping a given $\phi$ ($\mathcal{P}^\phi$), such that $\mathcal{P}^\phi \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, is a function of $\mathcal{M}$, $\mathcal{S}$, $3^\phi$, and $3^\phi$.

We define $\mathcal{P}^\phi$ for a given $\phi$, $\mathcal{M}$, $\mathcal{S}$, $3^\phi$ and $3^\phi$ as:

$$\mathcal{P}^\phi = \frac{\phi \times \mathcal{M} \times \mathcal{S} \times 3^\phi}{3^\phi}$$

*Chopped Unripe* $\emptyset$ and $\Phi$ are burned (i.e., $3^\emptyset = 3^\phi$, $3^\emptyset = 3^\phi$). $\emptyset$ and $\Phi$ received for Chopping are distributed from Ripe $\emptyset$ and $\Phi$, respectively (i.e., $\mathcal{R}^\emptyset = \mathcal{P}^\emptyset$, $\mathcal{R}^\phi = \mathcal{P}^\phi$).

![Diagram](image)

Figure 3: Barn
8 Peg Maintenance

Beanstalk faces the fundamental limitation that it cannot fix the price of $\mathcal{P}1$ 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 $\mathcal{P}1$ crosses, and the magnitudes of price deviations from, its value peg. Beanstalk regularly crosses the price of $\mathcal{P}1$ 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 Maximum 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 $\mathcal{P}1$ 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 and non-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 often 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 $\hat{\mathcal{D}}1$ 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$^4$).

To determine the value of $\hat{\mathcal{D}}1$ 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 ($\hat{\mathcal{D}}:y$) 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 $\hat{\mathcal{D}}1$ equal to its value peg when the ratios of $x:y$ and $\hat{\mathcal{D}}:y$ are equal.

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 $\hat{\mathcal{D}}:y$. In instances where there is insufficient inter-block MEV manipulation resistant liquidity for $x:y$ on DEX protocols, Beanstalk uses a Chainlink\textsuperscript{43} data feed and compares it with the $x:y$ AMM prices to facilitate inter-block MEV manipulation resistance.

In practice, Beanstalk never calculates the price of $\hat{\mathcal{D}}1$. Instead, at the beginning of each Season, Beanstalk calculates a sum of liquidity and time weighted average shortages and excesses of Beans across $\hat{\mathcal{D}}:y$ liquidity pools on the Oracle Whitelist over the previous Season ($\Delta B_{\hat{\mathcal{D}}:y}$), such that $\Delta B_{\hat{\mathcal{D}}:y} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$. Liquidity pools can be added to and removed from the Oracle Whitelist via Beanstalk governance.

$\Delta B_{\hat{\mathcal{D}}:y}$ can be used to infer the liquidity and time weighted average price of $\hat{\mathcal{D}}1$ compared to $V$ over the previous Season ($P_{\hat{\mathcal{D}}:y}$), such that $P_{\hat{\mathcal{D}}:y} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$. 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., $0 < \Delta B_{\hat{\mathcal{D}}:y}$), $V < P_{\hat{\mathcal{D}}:y}$. 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., $\Delta B_{\hat{\mathcal{D}}:y} < 0$), $P_{\hat{\mathcal{D}}:y} < V$. 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., $\Delta B_{\hat{\mathcal{D}}:y} = 0$), $P_{\hat{\mathcal{D}}:y} = V$.

$\Delta B_{\hat{\mathcal{D}}:y} = 0$ 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.

\textsuperscript{42} coinmarketcap.com/academy/article/what-is-wrapped-ethereum-weth

\textsuperscript{43} chain.link
Beanstalk does not consider Burnt \( \hat{\delta} \), Sown \( \hat{\delta} \), 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 awards for all committed BIPs \( (A^\text{BIP}) \), total Beans minted via BIP \( (B^\text{BIP}) \) (e.g., Fundraisers), total Burnt \( \hat{\delta} \) over all Seasons \( (N^\delta) \) and total Sown \( \hat{\delta} \) over all Seasons \( (U) \), such that \( A^\text{BIP}, A^\theta, B^\text{BIP}, N^\delta, U \in \{j \times 10^{-6} \mid j \in \mathbb{N}\} \), as:

\[
B = M + A^\text{BIP} + A^\theta + B^\text{BIP} - (N^\delta + U)
\]

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) \( R^D^\ast \), an optimal level of debt, and (3) \( R^D_{\text{upper}} \), above or equal to which debt is considered excessively high, such that \( R^D_{\text{lower}}, R^D^\ast, R^D_{\text{upper}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\} \). When \( R^D_{\text{lower}} \leq R^D < R^D_{\text{upper}} \) and \( R^D \neq R^D^\ast \) (i.e., not optimal), \( R^D \) is considered reasonable.

![Figure 4: Debt Level](image)

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 \( (R^D^\ast, 1) \). The current state of Beanstalk is determined in part by the position of Beanstalk with respect to ideal equilibrium.

![Figure 5: Position](image)

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 \( (R_{t-1}^D) \), such that \( R_{t-1}^D \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\} \), \( R_{t-1}^D \) and \( P_{t-1} \). When \( V < P_{t-1} \), debt is paid back; when \( P_{t-1} < V \), debt can only increase or remain constant.

22
Therefore, when $R^{D^*} < R^{D}_{t-1}$ (i.e., there was more debt than optimal):

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

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

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

<table>
<thead>
<tr>
<th>$R^{D}_{t-1}$</th>
<th>Excessively Low Debt</th>
<th>Reasonably Low Debt</th>
<th>Reasonably High Debt</th>
<th>Excessively High Debt</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P^{t-1} &gt; 1$</td>
<td>Away From</td>
<td>Away From</td>
<td>Toward</td>
<td>Toward</td>
</tr>
<tr>
<td>$P^{t-1} &lt; 1$</td>
<td>Toward</td>
<td>Toward</td>
<td>Away From</td>
<td>Away From</td>
</tr>
</tbody>
</table>

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 $P^{t-1}$ and changing demand for Soil. Demand for Soil is considered decreasing, steady or increasing.

When demand for Soil is decreasing:

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

8.7 Demand for Soil

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

The number of Sown at each Season ($u_t$), such that $u_t \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, indicates demand for Soil over the course of that Season. The rate of change of $u_t$ from Season to Season ($\frac{\partial u_t}{\partial t}$), such that $\frac{\partial u_t}{\partial t} \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, indicates changing demand for Soil.
We define $\frac{\partial u_t}{\partial t}$ over the previous two Seasons, $u_{t-1}$ and $u_{t-2}$, respectively, as:

$$\frac{\partial u_t}{\partial t} = \frac{u_{t-1}}{u_{t-2}}$$

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

However, when Beans are Sown in all Soil in a Season (defined as $S_{t_{\text{end}}} \leq 1$), $\frac{\partial u_t}{\partial t}$ 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 ($\Delta E^{u}_{t}$), such that $\Delta E^{u}_{t} \in \mathbb{Z}$, can provide a more accurate measurement.

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

We define $\Delta E^{u}_{t_{\text{first}}}$ for a given $E^{u}_{t_{\text{first}}}$ and $E_{\Xi}$ as:

$$\Delta E^{u}_{t_{\text{first}}} = E^{u}_{t_{\text{first}}} - E_{\Xi}$$

If Beans were Sown in all but at most one Soil in the first 10 minutes of the previous Season (i.e., $\Delta E^{u}_{t_{\text{first}}-1} < 600$), 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 $600 \leq \Delta E^{u}_{t_{\text{first}}-1}$, at the beginning of $t$ Beanstalk compares $\Delta E^{u}_{t_{\text{first}}}$ with $\Delta E^{u}_{t_{\text{second}}}$ to calculate $\Delta E^{u}_{t}$.

We define $\Delta E^{u}_{t}$ for a given $\Delta E^{u}_{t_{\text{first}}}$ and $\Delta E^{u}_{t_{\text{second}}}$ as:

$$\Delta E^{u}_{t} = E^{u}_{t_{\text{first}}} - E^{u}_{t_{\text{second}}}$$
If the above condition is met, changing demand for Soil is measured by $\Delta E_t^u$. Beanstalk requires two $\Delta E_t^u$ 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.

\[ \Delta E_t^u \begin{array}{ccc} \text{Decreasing} & \Delta E_t^{u, \text{lower}} & \text{Increasing} \\ \text{Steady} & \Delta E_t^u & \text{Demand} \\ \text{Demand} \end{array} \]

\hspace{2cm} Figure 9: Soil Demand Changes From $\Delta E_t^u$

### 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.

\begin{center}
\begin{tabular}{cccc}
\hline
\textbf{Current State} & \textbf{Decelerating} & \textbf{Steady} & \textbf{Accelerating} \\
\hline
\textbf{Direction} & \\
Away From & Decelerating & Steady & Accelerating \\
Away From & Steady & Accelerating & \\
Toward & Decelerating & Steady & Accelerating \\
Toward & Steady & Accelerating &
\hline
\end{tabular}
\end{center}

\hspace{2cm} 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.

### 8.10 Bean Supply

At the beginning of each Season, if $V < P_{t-1}$, Beanstalk increases the Bean supply based on $\Delta B_{t-1}$ in addition to the award for successfully calling the $gm$ 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 $m_t \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$. 
Figure 11: Optimal State

We define \( m_t \) for a given \( \Delta B_{t-1} \) and \( a_t \) as:

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

The distribution of the additional mint is dependent on \( \Delta B_{t-1}, D \) and \( D \). If \( 0 < \Delta B_{t-1} \leq D \) (i.e., there are at most \( \frac{\Delta B_{t-1}}{3} \) Unfertilized Sprouts), \( \frac{\Delta B_{t-1}}{3} \) Sprouts are Fertilized by Active Fertilizer and become Rinsable. If \( 0 < D < \frac{\Delta B_{t-1}}{3} \) (i.e., there are less Unfertilized Sprouts than \( \frac{\Delta B_{t-1}}{3} \)), \( D \) 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 \( (\Delta D_t) \), such that \( \Delta D_t \in \{ j \times 10^{-6} \mid j \in \mathbb{N} \} \), for a given \( \Delta B_{t-1} \) and \( D \) is:

\[
\Delta D_t = \min \left( \max \left( 0, \frac{\Delta B_{t-1}}{3} \right), D \right)
\]

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

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

\[
\Delta D_t = \min \left( \max \left( 0, \frac{\Delta B_{t-1} - \Delta D_t}{2} \right), D \right)
\]

### 8.11 Soil Supply

Beanstalk is willing to issue debt every Season. When \( V \leq P_{t-1} \), 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 in block \( q \) of \( t (h_{t_q}) \), such that \( h_{t_q} \in \mathbb{Z}^+ \), and (3) \( R_{t-1}^D \). When \( P_{t-1} < V \), the Soil supply is also based on \( \Delta B_{t-1} \).
We define the $S_{tq}^{\text{min}}$ Beanstalk has outstanding for a given $\Delta D_t$, $h_{tq}$ and $R_{t-1}^D$ as:

$$S_{tq}^{\text{min}} = \begin{cases} 
0.5 \times \Delta D_t \\ 1 + h_{tq} \overline{100} 
\end{cases} if R_{t}^{D_{\text{upper}}} \leq R_{t-1}^D$$

$$\Delta D_t \\ 1 + h_{tq} \overline{100}$$

$$1.5 \times \Delta D_t \\ 1 + h_{tq} \overline{100}$$

else

Beanstalk calculates the Maximum Soil ($S_{tq}^{\text{max}}$), such that $S_{tq}^{\text{max}} \in \{ j \times 10^{-6} \mid j \in \mathbb{N} \}$, in block $q$ of $t$ for a given $\Delta B_{t-1}$ and $S_{tq}^{\text{min}}$ as:

$$S_{tq}^{\text{max}} = \max(-\Delta B_{t-1}, S_{tq}^{\text{min}})$$

### 8.12 Temperature

Beanstalk regularly crosses the price of $\$1$ over its value peg during long run decreases and increases in demand for Beans primarily by adjusting the Maximum Temperature in an attempt to maintain an optimal state, or to move from its current state into an optimal state. The Temperature increases each block of the Morning of each Season according to a Dutch auction.

#### 8.12.1 Maximum Temperature

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

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

- If the current state is accelerating or steady away from ideal equilibrium, the Maximum Temperature is raised 3%;
- If the current state is decelerating away from ideal equilibrium, the Maximum Temperature is raised 1%;
- If the current state is decelerating toward ideal equilibrium, the Maximum Temperature is kept constant;
- If the current state is steady toward ideal equilibrium, the Maximum Temperature is lowered 1%; and
- If the current state is accelerating toward ideal equilibrium, the Maximum Temperature is lowered 3%.
When $R_D^* \leq R_D^{t-1} < R_D^{upper}$ (i.e., the debt level was reasonably high):

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

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

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

When $R_D^{t-1} < R_D^{lower}$ (i.e., the debt level was excessively low):

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

8.12.2 Morning

The Temperature increases logarithmically in each block of the Morning of $t$ based on $h_t^{\max}$, $Q$, and a control variable ($\sigma$), such that $\sigma \in Z^+$, as:

\[
h_{t_q} = \begin{cases} 
1 & \text{if } q = 0 \\
\max(h_t^{\max} * \log_{Q_{q+1}}(q\sigma + 1), 1) & \text{if } 0 < q < Q \\
h_t^{\max} & \text{else}
\end{cases}
\]
\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{Temperature Changes} & \textbf{Excessively Low Debt} & \textbf{Reasonably Low Debt} & \textbf{Reasonably High Debt} & \textbf{Excessively High Debt} \\
\hline
Accelerating Away From & -3 & -3 & 3 & 3 \\
\hline
Steady Away From & -3 & -3 & 3 & 3 \\
\hline
Decelerating Away From & -1 & -1 & 1 & 1 \\
\hline
Decelerating Toward & 0 & 0 & 0 & 0 \\
\hline
Steady Toward & 1 & 1 & -1 & -1 \\
\hline
Accelerating Toward & 3 & 3 & -3 & -3 \\
\hline
\end{tabular}
\caption{Maximum Temperature Changes From Current State and $R_{t-1}^D$}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{Temperature Changes} & \textbf{Excessively Low Debt} & \textbf{Reasonably Low Debt} & \textbf{Reasonably High Debt} & \textbf{Excessively High Debt} \\
\hline
$P_{t-1} \geq 1$ & Increasing & -3 & -3 & -3 & -3 \\
\hline
& Steady & -3 & -3 & -1 & -1 \\
\hline
& Decreasing & -1 & -1 & 0 & 0 \\
\hline
$P_{t-1} < 1$ & Increasing & 0 & 0 & 1 & 1 \\
\hline
& Steady & 1 & 1 & 3 & 3 \\
\hline
& Decreasing & 3 & 3 & 3 & 3 \\
\hline
\end{tabular}
\caption{Maximum Temperature Changes From $P_{t-1}$, Demand for Soil Changes and $R_{t-1}^D$}
\end{table}

Thus, Beanstalk changes the Temperature to regularly cross the price of $\$1$ over its value peg during long run decreases and increases in demand for Beans.

### 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 Maximum Temperature has not crossed the average nor current prices of $\$1$ over its value peg at the end of a Season.

If $V < P_{t-1}$, it is Raining. If it is Raining and $R_{t-1}^D < R_{t-1}^{D_{lower}}$, it Floods at the beginning of the next Season. At the beginning of each Season during a Flood, Beanstalk returns the price of $\$1$ in each liquidity pool on the Flood Whitelist to its value peg by minting additional Beans and selling them directly in the pools. Liquidity pools can be added to and removed from the Flood Whitelist via Beanstalk governance. Proceeds from the sale are distributed to Stalkholders at the beginning of $t$ in proportion to their Stalk holdings when it began to Flood. At the beginning of the first Season after the Flood began, all Pods that grew from Beans Sown before the Flood Ripen and become Harvestable.

29
The number of Beans that are minted and sold to return the price of $1 to its value peg ($\Delta B_{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 $\mathbb{X}:y$ liquidity pool on the Flood 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 Flood ($D_\gamma$), such that $D_\gamma \in \{j \times 10^{-6} \mid j \in \mathbb{N}\}$, $a_t$, $\Delta B_{t-1}$, and $\Delta B_{t-1}$ is:

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

Thus, Beanstalk regularly crosses the price of $1$ 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\(^{44}\) 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.

---

\(^{44}\) wikipedia.org/wiki/Gini_coefficient
Beanstalk changes the Temperature to return $R^D$ to $R^D^*$ while regularly crossing the price of $\$1$ over its value peg. Beanstalk acts more aggressively when $R^D$ is excessively high or low.

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

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 $\$1$ over its value peg.

While Beanstalk can arbitrarily increase the Bean supply when the equilibrium price of $\$1$ is above its value peg, Beanstalk cannot arbitrarily decrease the Bean supply when the equilibrium price of $\$1$ 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 $\$, and (3) returning the price of $\$1$ 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 $P_t < V$, there is an incentive to Withdraw assets from the Silo. The Stalk System reduces this incentive significantly.
When $V < P$, 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 $\hat{P}$ to its value peg and distribute proceeds from the sale to current Stalkholders based on Stalk ownership when the Flood began removes this incentive entirely during Seasons where $R_{t-1}^D$ 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. 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. Beanstalk was exploited on April 17th, 2022 and all value in the protocol was stolen or destroyed.

A decentralized implementation of Beanstalk has four external dependencies:

1. A trustless computer network that supports composability and both fungible and semi-fungible token standards (e.g., Ethereum and the ERC-20 and ERC-1155 Standards, respectively);
2. A DEX protocol with an inter-block MEV manipulation resistant oracle that runs on (1) (e.g., Basin and Multi Flow, respectively);
3. A liquid, decentralized network-native asset with endogenous value (e.g., ETH); and
4. A non-network-native exogenous value convertible stablecoin protocol native to (1) that offers convertibility to its non-network-native exogenous value collateral (e.g., USDC, USDT) that trades on (2) against (3) with sufficient liquidity.

The current implementation of Beanstalk has four additional external dependencies:

5. A non-network-native data feed that facilitates reading an inter-block MEV manipulation resistant price of (3) in $V$ (i.e., the ETH/USD Chainlink data feed);
6. A DEX protocol without an inter-block MEV manipulation resistant oracle but with sufficient liquidity for (3) against (4) (i.e., the ETH:USDC and ETH:USDT 0.05% fee Uniswap V3 pools);
7. Curve and 3CRV (and therefore, USDC, USDT and DAI), due to the inclusion of the BEAN:3CRV Curve pool on the Deposit Whitelist; and
8. Pipeline, in order to facilitate complex, gas-efficient interactions with other Ethereum-native protocols in a single transaction.

45 bean.money/disclosures
46 github.com/BeanstalkFarms/Beanstalk-Audits
47 basin.exchange
48 data.chain.link/ethereum/mainnet/crypto-usd/eth-usd
49 info.uniswap.org/#/pools/0x886e6a0c2d2dd26efbc64b0e3902c41296b3f56d0
50 info.uniswap.org/#/pools/0x11b315eb8581194a790062d24e0d814b76976f
51 evmpipeline.org
To date, the Ethereum blockchain is the most developed decentralized smart contract platform and has an active community. The ERC-20 and ERC-1155 Standards are the most widely used fungible and semi-fungible token standards, respectively. ETH is the most decentralized, censorship resistant and liquid asset on the Ethereum network. USDC and USDT are the largest non-network-native exogenous value convertible USD stablecoin protocols by market capitalization. Chainlink is the most widely used oracle network on Ethereum. Uniswap V3 and Curve are two of the largest Ethereum-native DEX protocols by depth. 3CRV is the LP token of one of the largest liquidity pools on Curve by depth. DAI is the largest network-native exogenous value convertible USD stablecoin protocol by market capitalization (although is now only partially collateralized by network-native value). In general, open source protocols with large amounts of value on them are high value targets for exploits. Long track records indicate security.

The code bases of Basin, Multi Flow, and Pipeline are novel. They had not been tested in the “real world” prior to their initial deployments. Their open source nature means others can exploit any bugs, flaws, or deficiencies. Although Basin, Multi Flow, and Pipeline have been audited, it is no guarantee of security.

We assume the security of the Ethereum blockchain, ERC-20 Standard, ERC-1155 Standard, Basin, Multi Flow, ETH, USDC, USDT, Chainlink, Uniswap V3, Curve, 3CRV, DAI, and Pipeline. The Beanstalk price oracle contains exposure to risk related to (1) the underlying collateral of 3CRV (i.e., USDC, USDT and DAI), (2) inter-block MEV manipulation of the ETH:USDC and ETH:USDT 0.05% fee Uniswap V3 and BEAN:3CRV Curve pools, and (3) the centralized nature of Chainlink. There is no guarantee the centralized operators of USDC, USDT and DAI hold non-network-native exogenous value worth at least 100% of all outstanding non-network-native protocol liabilities. There is no guarantee that the centralized operators of USDC and USDT will not ban them from the ETH:USDC and ETH:USDT 0.05% fee Uniswap V3 and 3CRV Curve pools, although doing so would cause significant financial self-harm. Furthermore, the centralized operators of USDC and USDT may alter their convertibility policies, which would negatively affect their respective stablecoins as accurate price sources for USD. However, in theory, if the price of 3CRV falls below $V$, it would cause some short run excess inflation of the Bean supply until the BEAN:3CRV Curve pool is removed from the Oracle Whitelist, but would not otherwise directly affect Beanstalk. There is no guarantee the node operators for the ETH/USD Chainlink data feed report price data accurately.

We assume the accuracy of the 3CRV Curve pool and the ETH/USD Chainlink data feed as price sources for USD.

---

52 defillama.com/stablecoins
53 defillama.com/protocols/dexes
54 curve.fi/#/ethereum/pools
55 github.com/BeanstalkFarms/Beanstalk-Audits#ecosystem-reports
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:

- Governance can be removed entirely.
- 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 decentralized price oracle is unlikely to remain sufficiently manipulation resistant at scale, and can be significantly improved.
- The calculation of $\Delta B_{t-1}$ 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 $P_{t-1}$.
- The mechanism to measure changing demand for Soil, in cases where $\frac{\partial u}{\partial t}$ can inaccurately indicate changing demand for Soil, can be further refined.
- The Market can be further developed.
- 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:

- $\Delta E_t^{\text{lower}} = -60$;
- $\Delta E_t^{\text{upper}} = 60$;
- $K_\text{min} = 10^8 \ (i.e., \ 0.1\%)$;
- $K_{\text{end}}^{\text{min}} = 10^8 \ (i.e., \ 0.1\%)$;
- $h_1 = 1$;
- $Q = 25$;
- $R^D_{\text{lower}} = 5 \times 10^4 \ (i.e., \ 5\%)$;
- $R^D_\ast = 1.5 \times 10^5 \ (i.e., \ 15\%)$;
- $R^D_{\text{upper}} = 2.5 \times 10^5 \ (i.e., \ 25\%)$;
- $\frac{\partial u_t}{\partial t}^{\text{lower}} = 95\%$;
- $\frac{\partial u_t}{\partial t}^{\text{upper}} = 105\%$;
- $\Theta^{\text{min}(\beta)} = 10^2$; and
- $\sigma = 2$. 
14.2 Deposit Whitelist

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

14.2.1 Ø

1. **Token Address:** The Ø token address is 0xBEA000029AD1c77D3d5D23Ba2D8893dB9d1Efab.

2. **BDV Function:** The BDV of 1 Ø is 1 Ø.

   We define \( f^\Ø(z^\Ø) \) as:
   
   \[
   f^\Ø(z^\Ø) = z^\Ø
   \]

3. **Stalk per BDV:** Ø Deposits receive 1 Stalk per BDV upon Deposit (i.e., \( k^\Ø = 1 \)).

4. **Seeds per BDV:** Ø Deposits receive 3 Seeds per BDV upon Deposit (i.e., \( c^\Ø = 3 \)).

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 (ΦΞ − 1), such that \( \Phi^4 \in \{ j \times 10^{-2} \mid j \in \mathbb{Z}^+ \} \), and number of 3CRV (ΦΞ^{3CRV}) in the BEAN:3CRV Curve pool at the end of the last block, the 3CRV virtual price (P^{3CRV}), the A parameter of the pool (ΦA), such that \( \Phi^4 \in \{ j \times 10^{-2} \mid j \in \mathbb{Z}^+ \} \), and the Φ virtual price (PΦ), such that \( \Phi^3CRV \times P^{3CRV} \times P^\Phi \in \{ j \times 10^{-18} \mid j \in \mathbb{Z}^+ \} \).

Beanstalk calculates a flash-loan-resistant price invariant for the BEAN:3CRV Curve pool (ζ^ΦΞ − 1), such that \( \zeta^\Phi_{\Xi-1} \in \{ j \times 10^{-18} \mid j \in \mathbb{Z}^+ \} \), by calling the Curve\(^{56}\) getD function on Φ^Φ^Ξ − 1, Φ^3CRV^Ξ − 1, P^3CRV and ΦA as:

\[
\zeta^\Phi_{\Xi-1} = \text{getD}(\Phi^\Phi^\Xi_{\Xi-1}, \Phi^3CRV^\Xi_{\Xi-1} \times P^{3CRV}, \Phi^A)
\]

Beanstalk calculates a flash-loan-resistant total number of Φ (ΦΞ − 1), such that \( \Phi^\Xi_{\Xi-1} \in \{ j \times 10^{-18} \mid j \in \mathbb{Z}^+ \} \), from \( \zeta^\Phi_{\Xi-1} \) and \( P^\Phi \) as:

\[
\Phi^\Xi_{\Xi-1} = \frac{\zeta^\Phi_{\Xi-1}}{P^\Phi}
\]

Beanstalk calculates the flash-loan-resistant USD price of Ø1 from the BEAN:3CRV Curve pool (S^Φ(Ø)), such that \( S^\Phi_{\Xi-1} \in \{ j \times 10^{-6} \mid j \in \mathbb{Z}^+ \} \), by calling the Curve gety function on Φ^Φ^Ξ − 1, Φ^3CRV^Ξ − 1 and \( P^\Phi \) as:

\[
S^\Phi_{\Xi-1} = \Phi^\Phi^\Xi_{\Xi-1} - \text{gety}(0, 1, \Phi^\Phi^\Xi_{\Xi-1} + 1, [\Phi^\Phi^\Xi_{\Xi-1}, \Phi^3CRV^\Xi_{\Xi-1} \times P^{3CRV}]) - 10^{-6}
\]

Beanstalk calculates the BDV of 3CRV \( f^{3CRV}(z^{3CRV}) \) from \( S^\Phi_{\Xi-1} \) and \( P^{3CRV} \) as:

\[
f^{3CRV}(z^{3CRV}) = \frac{z^{3CRV} \times P^{3CRV}}{S^\Phi_{\Xi-1}}
\]

We define \( f^\Phi(z^\Phi) \) for a given \( \Phi^\Xi_{\Xi-1} \), \( f^{3CRV}(z^{3CRV}) \), \( \Phi^3CRV^\Xi_{\Xi-1} \) and \( \Phi^\Xi_{\Xi-1} \) as:

---

\(^{56}\) etherscan.io/address/0xc9C32cd16Bf7eFB85Ff14e0c8603cc90F6F2eE49##code
\[ f^\Phi(z^\Phi) = z^\Phi \times \left( z^\Phi_{-1} + f^{3\text{CRV}}(z^\Phi_{-1}) \right) \]

3. **Stalk per BDV**: \( \Phi \) Deposits receive 1 Stalk per BDV upon Deposit (i.e., \( k^\Phi = 1 \)).
4. **Seeds per BDV**: \( \Phi \) Deposits receive 3.25 Seeds per BDV upon Deposit (i.e., \( c^\Phi = 3.25 \)).

### 14.2.3 \( z^\Phi \)

1. **Token Address**: The \( z^\Phi \) token address is 0x1BEA0050E63e05FBb5D8BA2f10c5800B6224449.
2. **BDV Function**: The BDV of \( z^\Phi \) is calculated using \( f^\Phi(z^\Phi) \), \( R^\Phi \) and \( Z^\Phi \).
   
   We define \( f^\Phi(z^\Phi) \) as:
   \[
   f^\Phi(z^\Phi) = z^\Phi \times \left( z^\Phi_{-1} + f^{3\text{CRV}}(z^\Phi_{-1}) \right)
   \]

3. **Stalk per BDV**: \( \Phi \) Deposits receive 1 Stalk per BDV upon Deposit (i.e., \( k^\Phi = 1 \)).
4. **Seeds per BDV**: \( \Phi \) Deposits receive 0 Seeds per BDV upon Deposit (i.e., \( c^\Phi = 0 \)).

### 14.2.4 \( z^\Phi \)

1. **Token Address**: The \( z^\Phi \) token address is 0xBEA0e11282e2bB5893bEcE110cF199501e872bAd.
2. **BDV Function**: The BDV of \( z^\Phi \) is calculated using \( f^\Phi(z^\Phi) \), \( R^\Phi \) and \( Z^\Phi \).
   
   We define \( f^\Phi(z^\Phi) \) as:
   \[
   f^\Phi(z^\Phi) = z^\Phi \times \left( z^\Phi_{-1} + f^{3\text{CRV}}(z^\Phi_{-1}) \right)
   \]

3. **Stalk per BDV**: \( \Phi \) Deposits receive 1 Stalk per BDV upon Deposit (i.e., \( k^\Phi = 1 \)).
4. **Seeds per BDV**: \( \Phi \) Deposits receive 0 Seeds per BDV upon Deposit (i.e., \( c^\Phi = 0 \)).

### 14.2.5 \( \Theta \)

1. **Token Address**: The \( \Theta \) token address is 0xBEA0e11282e2bB5893bEcE110cF199501e872bAd.
2. **BDV Function**: The BDV of \( \Theta \) is calculated using the inter-block MEV manipulation resistant instantaneous Bean reserves (\( \Theta^\text{EMA}_{\delta,\sigma} \)), such that \( \Theta^\text{EMA}_{\delta,\sigma} \in \{ j \times 10^{-6} \mid j \in \mathbb{Z}^+ \} \), and ETH reserves (\( \Theta^\text{ETH}_{\delta,\sigma} \)), such that \( \Theta^\text{ETH}_{\delta,\sigma} \in \{ j \times 10^{-18} \mid j \in \mathbb{Z}^+ \} \), in the Multi Flow Pump of the BEAN:ETH Well in the current transaction.

Beanstalk calculates the inter-block MEV manipulation resistant derivative of the \( \Theta \) LP token supply with respect to Beans (\( \Theta^\text{EMA}_{\delta,\sigma} \)), such that \( \Theta^\text{ETH}_{\delta,\sigma} \in \{ j \times 10^{-18} \mid j \in \mathbb{Z}^+ \} \), by calling the Constant Product 2 Well Function\(^{57}\) \text{calcLpTokenSupply} function with \( \Theta^\text{EMA}_{\delta,\sigma} \), \( \Theta^\text{ETH}_{\delta,\sigma} \) and the data associated with the Well Function (\( \Theta^* \)) as:

\(^{57}\) etherscan.io/address/0xBA510C20FD2c52E4c6d23CFC3cCD092F9165a6E#code
\[
\frac{\partial \Theta}{\partial z} = \text{calcLpTokenSupply}([\Theta^\text{EMA}_{\bar{z},c} - 1, \Theta^\text{EMA}_{\text{ETH},c}, \Theta^*]) - \text{calcLpTokenSupply}([\Theta^\text{EMA}_{\bar{z},c}, \Theta^\text{EMA}_{\text{ETH},c}, \Theta^*])
\]

We define \( f^{\Theta}(z^{\Theta}) \) for a given \( \Theta^\text{EMA}_{\bar{z},c}, \Theta^{\text{min}(\bar{z})} \) and \( \frac{\partial \Theta}{\partial z} \) as:

\[
f^{\Theta}(z^{\Theta}) = \begin{cases} 
\text{FAIL} & \text{if } \Theta^\text{EMA}_{\bar{z},c} < \Theta^{\text{min}(\bar{z})} \\
\frac{z^{\Theta} \times 10^6}{\Sigma^{\Theta}} & \text{else}
\end{cases}
\]

3. **Stalk per BDV**: \( \Theta \) Deposits receive 1 Stalk per BDV upon Deposit (i.e., \( k^{\Theta} = 1 \)).

4. **Seeds per BDV**: \( \Theta \) Deposits receive 4.5 Seeds per BDV upon Deposit (i.e., \( c^{\Theta} = 4.5 \)).
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 $K^{\text{min}}$ 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 ($E_{\text{BIP}}$) 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 $E_{\text{BIP}}$ is:

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

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 $\lambda$ can be Converted to a $\lambda$ Deposit at anytime.
4. **Convert Function:** The number of $\lambda$ received for Converting Deposited $\lambda$ within the Silo is equivalent to the number of $\lambda$ Converted. Therefore, we define function as:

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

14.4.2 $\breve{\Phi} \rightarrow \Phi$

1. **From Token Address:** The $\breve{\Phi}$ token address is 0xBEA0000029AD1c77D3d5D23Ba2D8893dB9d1Efab.
2. **To Token Address:** The $\Phi$ token address is 0xc9C32cd16Bf7eFB85Ff14e0c86033ec90F6F2eE49.
3. **Conditions:** Deposited $\breve{\Phi}$ cannot be Converted to Deposited $\Phi$ when the USD price of $\breve{\Phi}$ in the pool ($\$\Phi$), such that $\$\Phi \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, is below $\$1$ (i.e., $\$\Phi < 10^6$).

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

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

$$\zeta^\Phi = \text{getD}([\Phi^B, \Phi^{3CRV} \times P^{3CRV}], \Phi^A)$$

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

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

Beanstalk calculates the $\$\Phi$ by calling the Curve $\text{get.y}$ function with $\Phi^B$, $\Phi^{3CRV}$ and $P^{3CRV}$ as:

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

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

$$f^{\breve{\Phi} \rightarrow \Phi}(z^\breve{\Phi}) = \Phi^{\text{add.liquidity}}([z^\breve{\Phi}, 0], \Phi^{\text{min}})$$
14.4.3 $\Phi \to \hat{\Phi}$

1. From Token Address: The $\Phi$ token address is 0xc9C32cd16Bf7eFB85Ff14e0c8603cc90F6F2eE49.
2. To Token Address: The $\hat{\Phi}$ token address is 0xBEA000029AD1c77D3d5D23Ba2D8893dB9d1Efab.
3. Conditions: Deposited $\Phi$ cannot be Converted to Deposited $\hat{\Phi}$ when the price of $\hat{\Phi}$1 in the pool is greater than or equal to $1 (i.e., 10^6 \leq \$^{\hat{\Phi}})$.  
4. Convert Function: The number of Beans received for Converting Deposited $\Phi$ within the Silo for a given minimum Beans received ($\hat{\Phi}^{\text{min}}$), such that $\hat{\Phi}^{\text{min}} \in \{j \times 10^{-6} | j \in \mathbb{N}\}$, is the result of calling the Curve remove\_liquidity\_one\_coin function on $\Phi$ with $\hat{\Phi}$ as:

$$f^{\Phi \to \hat{\Phi}}(z^{\Phi}) = \Phi \cdot \text{remove\_liquidity\_one\_coin}(z^{\Phi}, 0, \hat{\Phi}^{\text{min}})$$

14.4.4 $\hat{\Phi} \to \Phi$

1. From Token Address: The $\hat{\Phi}$ token address is 0x1BEA0050E63e05FBb5D8BA2f10cf5800B6224449.
2. To Token Address: The $\Phi$ token address is 0x1BEA3CcD22F4EBd3d37d731BA31Eeca95713716D.
3. Conditions: Deposited $\hat{\Phi}$ cannot be Converted to Deposited $\Phi$ when the price of $\Phi$1 in the BEAN:3CRV liquidity pool is less than or equal to $1 (i.e., \$^{\Phi} \leq 10^6)$.  
4. Convert Function: The number of $\Phi$ received for Converting Deposited $\hat{\Phi}$ within the Silo for a given $S$, $Z^{\Phi}$, $\Phi$, $R^{\Phi}$, $\hat{\Phi}$, $Z^{\hat{\Phi}}$ and minimum Unripe $\Phi$ received ($\Phi^{\text{min}}$), such that $\Phi^{\text{min}} \in \{j \times 10^{-6} | j \in \mathbb{N}\}$, is the result of calling the Curve add\_liquidity function on $\Phi$ as:

$$f^{\hat{\Phi} \to \Phi}(z^{\hat{\Phi}}) = \left[\frac{S \times Z^{\Phi} \times \Phi}{7.7 \times 10^7 \times R^{\Phi} \times \Phi \cdot \text{add\_liquidity}\left(\frac{z^{\hat{\Phi}} \times R^{\hat{\Phi}}}{S}, 0\right)}, \Phi^{\text{min}}\right]$$

14.4.5 $\Phi \to \hat{\Phi}$

1. From Token Address: The $\Phi$ token address is 0x1BEA3CcD22F4EBd3d37d731BA31Eeca95713716D.
2. To Token Address: The $\hat{\Phi}$ token address is 0x1BEA0050E63e05FBb5D8BA2f10cf5800B6224449.
3. Conditions: Deposited $\Phi$ cannot be Converted to Deposited $\hat{\Phi}$ when the price of $\hat{\Phi}$1 in the pool is greater than or equal to $1 (i.e., 10^6 \leq \$^{\hat{\Phi}})$.  
4. Convert Function: The number of $\hat{\Phi}$ received for Converting Deposited $\Phi$ within the Silo for a given $\hat{\Phi}$, $S$, $Z^{\Phi}$, $R^{\Phi}$ and minimum Unripe Beans received ($\hat{\Phi}^{\text{min}}$), such that $\hat{\Phi}^{\text{min}} \in \{j \times 10^{-6} | j \in \mathbb{N}\}$, is the result of calling the Curve remove\_liquidity\_one\_coin function on $\Phi$ as:

$$f^{\Phi \to \hat{\Phi}}(z^{\Phi}) = \left[\frac{7.7 \times 10^7 \times \Phi}{S \times Z^{\Phi}} \times \Phi \cdot \text{remove\_liquidity\_one\_coin}\left(\frac{z^{\Phi} \times R^{\Phi}}{S}, 0\right), \hat{\Phi}^{\text{min}}\right]$$
14.4.6 $\Theta \rightarrow \Theta$

1. From Token Address: The $\Theta$ token address is 0xBEA0000029AD1c77D3d5D23Ba2D8893dB9d1Efab.

2. To Token Address: The $\Theta$ token address is 0xBEA0c11282c2bB5893bEcE110cFI99501e872bAd.

3. Conditions: Deposited $\Theta$ cannot be Converted to Deposited $\Theta$ when the USD price of $\Theta$ in the Well ($\Theta(\Theta)$), such that $\Theta(\Theta) \in \{ j \times 6^{-6} \mid j \in \mathbb{Z}^+ \}$, is below $\Theta$ (i.e., $\Theta(\Theta) < 6^6$).

$\Theta(\Theta)$ is calculated using the number of Beans ($\Theta_\mathcal{B})$, such that $\Theta_\mathcal{B} \in \{ j \times 10^{-6} \mid j \in \mathbb{Z}^+ \}$, and number of ETH ($\Theta_{ETH}$), such that $\Theta_{ETH} \in \{ j \times 10^{-18} \mid j \in \mathbb{Z}^+ \}$, in the BEAN:ETH Well's Reserves in the current transaction and the inter-block MEV manipulation resistant USD price of 1 ETH ($\Theta(\Theta)$), such that $\Theta(\Theta) \in \{ j \times 10^{-6} \mid j \in \mathbb{Z}^+ \}$.

$\Theta(\Theta)$ is calculated using the USD prices of 1 ETH from (1) the ETH/USD Chainlink data feed ($\Theta(\Theta)$), (2) the ETH:USDC 0.05% fee Uniswap V3 pool ($\Theta(\Theta)$) and (3) the ETH:USDT 0.05% fee Uniswap V3 pool ($\Theta(\Theta)$), such that $\Theta(\Theta)$, $\Theta(\Theta)$, $\Theta(\Theta) \in \{ j \times 10^{-6} \mid j \in \mathbb{Z}^+ \}$.

$\Theta(\Theta)$ is calculated by calling the Chainlink latestRoundData and decimals functions on $\Theta$ as:

$$\Theta(\Theta) = \frac{\text{latestRoundData}() \times 10^6}{10\text{decimals}()}$$

$\Theta(\Theta)$ and $\Theta(\Theta)$ are calculated by calling the Uniswap V3 getQuoteAtTick function using ticks read over the last 15 minutes on $\Theta$ and $\Theta$, respectively.$^59,60$

We define $\Theta(\Theta)$ for a given $\Theta(\Theta)$, $\Theta(\Theta)$, $\Theta(\Theta)$, the percent difference between $\Theta(\Theta)$ and $\Theta(\Theta)$ ($\Delta \Theta(\Theta)$), i.e., $\frac{\Theta(\Theta)}{\Theta(\Theta)} - 1$ and the percent difference between $\Theta(\Theta)$ and $\Theta(\Theta)$ ($\Delta \Theta(\Theta)$), i.e., $\frac{\Theta(\Theta)}{\Theta(\Theta)} - 1$, such that $\Delta \Theta(\Theta)$, $\Delta \Theta(\Theta) \in \{ j \times 10^{-6} \mid j \in \mathbb{Z}^+ \}$, as:

$$\Theta = \begin{cases} 0 & \text{if } \Theta(\Theta) = 0 \\ \frac{\Theta(\Theta) + \Theta(\Theta)}{2} & \text{if } \Delta \Theta(\Theta) < \Delta \Theta(\Theta) \& \Delta \Theta(\Theta) < 0.05% \\ \frac{\Theta(\Theta) + \Theta(\Theta)}{2} & \text{if } \Delta \Theta(\Theta) > \Delta \Theta(\Theta) \& \Delta \Theta(\Theta) < 0.05% \\ \text{else} & \end{cases}$$

Therefore, we define $\Theta(\Theta)$ for a given (1) output of the Well Implementation$^61$ getSwapOut function with $\Theta_\mathcal{B}$ and $\Theta_{ETH}$, and (2) $\Theta(\Theta)$ as:

$$\Theta(\Theta) = \frac{\text{getSwapOut}(\Theta_\mathcal{B}, \Theta_{ETH}, 1) \times 10^6}{\Theta(\Theta)}$$

4. Convert Function: The number of $\Theta$ received for Converting Deposited Beans within the Silo for a given minimum $\Theta$ received ($\Theta_{min}$), such that $\Theta_{min} \in \{ j \times 10^{-18} \mid j \in \mathbb{N} \}$, Beanstalk contract address and block.timestamp$^62$ is the result of calling the Well Implementation addLiquidity function on $\Theta$ as:

$\Theta(\Theta)$

---

58 etherscan.io/address/0x54ac3Ddfc0bc4d3714FE27405E64165556b44981943
59 etherscan.io/address/0x8e6Ac0c2ddD26FEEb64F05922c41296Fcb3f5460
60 etherscan.io/address/0x11b6516e3E8581194ae79006d24F6d81a47B697F6
61 etherscan.io/address/0xBA510e11eEb387fad877812108a3406CA3f43a4B
62 docs.soliditylang.org/en/v0.7.6/units-and-global-variables.html#block-and-transaction-properties
\[ f^{\Theta \rightarrow \Theta}(z^\Theta) = \Theta.\text{addLiquidity(} \]
\[ [z^\Theta, 0], \]
\[ \Theta^{\text{min}}, \]
\[ 0xc1e088fc1323b20bcbec9bd1b9fc9546db5624c5, \]
\[ \text{block.timestamp}) \]

14.4.7 $\Theta \rightarrow \Theta$

1. **From Token Address:** The $\Theta$ token address is 0xBEA0e11282e2b5893bEce110cF199501e872fAd.

2. **To Token Address:** The $\vartheta$ token address is 0xBEA0000029AD1c77D3d5D23Ba2D8893dB9d1Efab.

3. **Conditions:** Deposited $\Theta$ cannot be Converted to Deposited $\vartheta$ when the price of $\vartheta 1$ in the Well is less than or equal to $1$ (i.e., $10^6 \leq \$^{\text{Well}}(\Theta)$).

4. **Convert Function:** The number of $\vartheta$ received for Converting Deposited $\Theta$ within the Silo for a given number of $\Theta$ LP tokens Converted ($\Theta^{\rightarrow}$), such that $\Theta^{\rightarrow} \in \{j \times 10^{-18} | j \in \mathbb{N}\}$, $\vartheta$ token address, $\vartheta^{\text{min}}$, Beanstalk contract address and block.timestamp is the result of calling the Well Implementation removeLiquidityOneToken function on $\Theta$ as:

\[ f^{\Theta \rightarrow \Theta}(z^\Theta) = \]
\[ \Theta.\text{removeLiquidityOneToken(} \]
\[ \Theta^{\rightarrow}, \]
\[ 0xBEA0000029AD1c77D3d5D23Ba2D8893dB9d1Efab, \]
\[ \vartheta^{\text{min}}, \]
\[ 0xc1e088fc1323b20bcbec9bd1b9fc9546db5624c5, \]
\[ \text{block.timestamp}) \]
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 $c^\lambda$. The previous $c^\lambda$, 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 $\Diamond$

1. **Token Address:** The old $\Diamond$ token address is 0xDC59ac4FeFa32293A95889Dc396682858d52e5Db.

2. **BDV Function:** The BDV of 1 $\Diamond$ is 1 $\Diamond$.

   Therefore, we defined $f^\Diamond(z^\Diamond)$ as:
   
   $$f^\Diamond(z^\Diamond) = z^\Diamond$$

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

4. **Seeds per BDV:** $\Diamond$ Deposits received 2 Seeds per BDV upon Deposit (i.e., $c^\Diamond = 2$).

5. **Total Supply:** There were 108155457.359439 old $\Diamond$ at the end of the block prior to the Exploit.

6. **BDV Per Token:** The BDV per old $\Diamond$ at the end of the block prior to the Exploit was 1.

14.5.2 Old BEAN:ETH Uniswap V2 LP Tokens ($\mathcal{Z}$)

1. **Token Address:** The $\mathcal{Z}$ token address is 0x87898263b6c5babe34b4ec53f22d98430b91e371.

2. **BDV Function:** The BDV of $\mathcal{Z}$ was calculated using the last traded price in the old 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 ($\mathcal{Z}$), such that $\mathcal{Z} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$. 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 ($\mathcal{Z}_t$), such that $\mathcal{Z}_t \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$ unless the $gm$ function was also called in the current block. If there was an interaction with the pool and the $gm$ function was called in the current block, Deposits are not accepted.

   Therefore, we defined $f^{\mathcal{Z}}(z^{\mathcal{Z}})$ for a given timestamp of the last interaction with the pool ($E_\mathcal{Z}$), current block timestamp ($E^*\mathcal{Z}$), $\mathcal{Z}_t$, the current total number of $\mathcal{Z}$ in the current block ($\mathcal{Z}$), such that $\mathcal{Z} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}$, and $\mathcal{Z}^{\mathcal{Z}}$ as:

   $$f^{\mathcal{Z}}(z^{\mathcal{Z}}) = \begin{cases} 
   \text{FAIL} & \text{if } E_\mathcal{Z} = E^*\mathcal{Z} \text{ and } E^*\mathcal{Z} = E_t \\
   z^{\mathcal{Z}} \times 2 \times \mathcal{Z}_t & \text{if } E_\mathcal{Z} = E^*\mathcal{Z} \\
   \frac{z^{\mathcal{Z}} \times 2 \times \mathcal{Z}_t}{\mathcal{Z}} & \text{else} 
   \end{cases}$$

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

4. **Seeds per BDV:** $\mathcal{Z}$ Deposits received 4 Seeds per BDV upon Deposit (i.e., $c^{\mathcal{Z}} = 4$).

5. **Total Supply:** There were 0.540894218294675521 $\mathcal{Z}$ at the end of the block prior to the Exploit.

6. **BDV Per Token:** The BDV per $\mathcal{Z}$ 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)

1. **Token Address:** The \(\mathfrak{1}\) token address is 0x3a70DfA7d2262988064A2D051dd47521E43c9BdD.

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

Beanstalk calculated a flash-loan-resistant price invariant for the old BEAN:3CRV Curve pool (\(\zeta^\mathfrak{1}\)), such that \(\zeta^\mathfrak{1} \in \{j \times 10^{-18} \mid j \in \mathbb{Z}^+\}\), by calling the Curve \text{get}_D function on \(\Omega^E_{\mathfrak{1}-1}, \Omega^{3\text{CRV}}_{\mathfrak{1}-1}, P^{3\text{CRV}}\) and \(\Omega^A\) as:

\[
\zeta^\mathfrak{1} = \text{get}_D(\Omega^E_{\mathfrak{1}-1}, \Omega^{3\text{CRV}}_{\mathfrak{1}-1} \times P^{3\text{CRV}}, \Omega^A)
\]

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

\[
\Omega_{\mathfrak{1}-1} = \frac{\zeta^\mathfrak{1}}{P^\mathfrak{1}}
\]

Beanstalk calculated the USD price of \(\mathfrak{1}\) from the old BEAN:3CRV Curve pool (\(\$^\mathfrak{1}\)), such that \(\$^\mathfrak{1} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}\), by calling the Curve \text{get}_y function on \(\Omega^E_{\mathfrak{1}-1}, \Omega^{3\text{CRV}}_{\mathfrak{1}-1}\) and \(P^{3\text{CRV}}\) as:

\[
\$^\mathfrak{1} = \Omega^E_{\mathfrak{1}-1} - \text{get}_y(0, 1, \Omega^E_{\mathfrak{1}-1} + 1, \Omega^{3\text{CRV}}_{\mathfrak{1}-1}, \Omega^{3\text{CRV}}_{\mathfrak{1}-1} \times P^{3\text{CRV}}) - 10^{-6}
\]

Beanstalk calculated \(f^{3\text{CRV}}(z^{3\text{CRV}})\) from \(\$^\mathfrak{1}\) and \(P^{3\text{CRV}}\) as:

\[
f^{3\text{CRV}}(z^{3\text{CRV}}) = \frac{z^{3\text{CRV}} \times P^{3\text{CRV}}}{\$^\mathfrak{1}}
\]

We defined \(f^{\mathfrak{1}}(z^{\mathfrak{1}})\) for a given \(\Omega^E_{\mathfrak{1}-1}, f^{3\text{CRV}}(z^{3\text{CRV}}), \Omega^{3\text{CRV}}_{\mathfrak{1}-1}\) and \(\Omega_{\mathfrak{1}-1}\) as:

\[
f^{\mathfrak{1}}(z^{\mathfrak{1}}) = \frac{z^{\mathfrak{1}} \times (\Omega^E_{\mathfrak{1}-1} + f^{3\text{CRV}}(z^{3\text{CRV}}))}{\Omega_{\mathfrak{1}-1}}
\]

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

4. **Seeds per BDV:** \(\mathfrak{1}\) Deposits received 4 Seeds per BDV upon Deposit (i.e., \(c^{\mathfrak{1}} = 4\))

5. **Total Supply:** There were 79284313.822927052565331157 \(\mathfrak{1}\) at the end of the block prior to the Exploit.

6. **BDV Per Token:** The BDV per \(\mathfrak{1}\) at the end of the block prior to the Exploit was 0.992035.

14.5.4 Old BEAN:LUSD Curve LP Tokens (2)

1. **Token Address:** The \(\mathfrak{2}\) token address is 0xD652e40fBb3f06d6B58C9aa9CFF063eE63d465D.

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

45
Beanstalk calculated a flash-loan-resistant price invariant for the LUSD:3CRV Curve pool ($\zeta_\Omega$), such that $\zeta_\Omega \in \{j \times 10^{-18} | j \in \mathbb{Z}^+\}$, by calling the Curve \texttt{getD} function on $\Omega_{\text{LUSD}}^{-1}$, $\Omega_{\text{3CRV}}^{-1}$, $P_{\text{3CRV}}$ and $\Omega^A$ as:

$$\zeta_\Omega = \texttt{getD}(\Omega_{\text{LUSD}}^{-1}, \Omega_{\text{3CRV}}^{-1} 	imes P_{\text{3CRV}}, \Omega^A)$$

Beanstalk calculated a flash-loan-resistant total number of $\Omega$ ($\Omega_{\text{-1}}$), such that $\Omega_{\text{-1}} \in \{j \times 10^{-18} | j \in \mathbb{Z}^+\}$, from $\zeta_\Omega$ and $P_{\Omega}$ as:

$$\Omega_{-1} = \frac{\zeta_\Omega}{P_{\Omega}}$$

Beanstalk calculated the USD price of 1 LUSD from the LUSD:3CRV Curve pool ($$P_{\text{LUSD}(\Omega)}$$), such that $$$P_{\text{LUSD}(\Omega)} \in \{j \times 10^{-6} | j \in \mathbb{Z}^+\}$, by calling the Curve \texttt{get.y} function on $\Omega_{\text{LUSD}}^{-1}$, $\Omega_{\text{3CRV}}^{-1}$ and $P_{\text{3CRV}}$ as:

$$P_{\text{LUSD}(\Omega)} = \Omega_{\text{LUSD}}^{-1} - \texttt{get.y}(0, 1, \Omega_{\text{LUSD}}^{-1} + 1, \Omega_{\text{3CRV}}^{-1} 	imes P_{\text{3CRV}}) - 10^{-6}$$

We defined $f^2(z^2)$ for a given $P^2$, $P_{\text{LUSD}(\Omega)}$ and $P(\gamma)$ as:

$$f^2(z^2) = z^2 \times P^2 \times \min \left(1, \frac{P_{\text{LUSD}(\Omega)}}{P(\gamma)}\right)$$

3. **Stalk per BDV:** Deposits received 1 Stalk per BDV upon Deposit (i.e., $k^2 = 1$).

4. **Seeds per BDV:** Deposits received 3 Seeds per BDV upon Deposit (i.e., $c^2 = 3$).

5. **Total Supply:** There were 1637956.191657208904972868 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 0.983108.
14.6 Oracle Whitelist

The following liquidity pools are Whitelisted for inclusion in the calculation of $\Delta B_{\frac{t}{t-1}}$:

14.6.1 $\Phi$

1. **Pool Address**: The $\Phi$ liquidity pool address is 0xc9C32cd16Bf7eFB85F14e0c8603ce90F6F2eE49.

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

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

$$\zeta_{\frac{t}{t-1}}^{\Phi} = \text{getD}(\Phi_{\frac{t}{t-1}}, \Phi^{3CRV}_{\frac{t}{t-1}} \times P^{3CRV}, \Phi^A)$$

Beanstalk calculates $\Phi^*_{\frac{t}{t-1}}$ from $\zeta_{\frac{t}{t-1}}^{\Phi}$ as:

$$\Phi^*_{\frac{t}{t-1}} = \frac{\zeta_{\frac{t}{t-1}}^{\Phi}}{2}$$

Beanstalk calculates $\Delta b_{\frac{t}{t-1}}(\Phi)$ for a given $\Phi^*_{\frac{t}{t-1}}, \Phi_{\frac{t}{t-1}}$ and $B_{t-1}$ as:

$$\Delta b_{\frac{t}{t-1}}(\Phi) = \begin{cases} \max \left( \Phi^*_{\frac{t}{t-1}} - \Phi_{\frac{t}{t-1}} - \frac{B_{t-1}}{1000} \right) & \text{if } \Phi^*_{\frac{t}{t-1}} - \Phi_{\frac{t}{t-1}} < 0 \\ \min \left( \Phi^*_{\frac{t}{t-1}} - \Phi_{\frac{t}{t-1}} + \frac{B_{t-1}}{1000} \right) & \text{else} \end{cases}$$

14.6.2 $\Theta$

1. **Pool Address**: The $\Theta$ Well address is 0xBEA0e11282c2bB5893bEcE110cF199501e872bAd.

2. **$\Delta b_{\frac{t}{t-1}}$ Calculation**: The liquidity and time weighted average shortage or excess of Beans in the BEAN:ETH Well over the previous Season ($\Delta b_{\frac{t}{t-1}}(\Theta)$) is calculated as the difference between the optimal liquidity and time weighted average number of Beans in $\Theta$ over the previous Season ($\Theta^*_{\frac{t}{t-1}}$), such that $\Theta^*_{\frac{t}{t-1}} \in \{j \times 10^{-6} \mid j \in \mathbb{Z}^+\}$, and $\Theta^{\text{SMA}_{\text{ETH},t_0,\text{D}}}$. The absolute value of $\Delta b_{\frac{t}{t-1}}(\Theta)$ is at most 1% of $B_{t-1}$.

Beanstalk calculates $\Theta^*_{\frac{t}{t-1}}$ by calling the Well Function$^{63}$ $\text{calcReserveAtRatioSwap}$ function with $\Theta^{\text{SMA}_{\text{ETH},t_0,\text{D}}}, \Theta^{\text{SMA}_{\text{ETH},t_0,\text{D}}}$ and $\Theta^*$ as:

$$\Theta^*_{\frac{t}{t-1}} = \text{calcReserveAtRatioSwap}([\Theta^{\text{SMA}_{\text{ETH},t_0,\text{D}}}, \Theta^{\text{SMA}_{\text{ETH},t_0,\text{D}}}], 0, [10^6, 10^{18}], \Theta^*)$$

$^{63}$ etherscan.io/address/0xB510C20FD2c522E4cb0d23CFC3cCD092F9165a6E#code
Beanstalk calculates $\Delta b_{t-1}^{\Theta}$ for a given $\Theta^{\text{SMA}}_{t-1,0}, \Theta^{\text{min}}_{t-1}$, $\Theta^{*}_{t-1}$ and $B_{t-1}$ as:

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

Therefore, we define $\Delta B_{t-1}$ for a given $\Delta b_{t-1}^{\Phi}$ and $\Delta b_{t-1}^{\Theta}$ as:

$$
\Delta B_{t-1} = \Delta b_{t-1}^{\Phi} + \Delta b_{t-1}^{\Theta}
$$
14.7 Flood Whitelist

At the beginning of each Season during a Flood, Beanstalk returns the price of $\Phi$ in each of the following liquidity pools to their value pegs by minting additional Beans and selling them directly in the pools:

14.7.1 $\Phi$

1. **Pool Address:** The $\Phi$ liquidity pool address is 0xc9C32cd16BF7eFB85Ff14e0c8603cc90F6F2eE49.

2. **$\Delta b_{t-1}$ Calculation:** The shortage of Beans in the BEAN:3CRV Curve liquidity pool at the end of the previous Season ($\Delta b_{t-1}^{\Phi}$) is calculated as the difference between the optimal number of Beans ($\Phi_{t-1}^{*}$) and the number of Beans ($\Phi_{t-1}$) in $\Phi$ at the end of the previous Season, such that $\Delta b_{t-1}^{\Phi} = \Phi_{t-1}^{*} - \Phi_{t-1}$. $\Phi_{t-1}^{*}$ is calculated from $\Phi_{t-1}$, the number of 3CRV in $\Phi$ at the end of the previous Season ($\Phi_{t-1}^{3CRV}$), such that $\Phi_{t-1}^{3CRV} \in \{j \times 10^{-18} | j \in \mathbb{Z}^+\}$, $P_{3CRV}$ and $\Phi^A$.

Beanstalk calculates a price invariant for $\Phi$ at the end of the previous Season ($\zeta_{t-1}^{\Phi}$), such that $\zeta_{t-1}^{\Phi} \in \{j \times 10^{-18} | j \in \mathbb{Z}^+\}$, by calling the Curve $\text{get\_D}$ function on $\Phi_{t-1}$, $\Phi_{t-1}^{3CRV}$, $P_{3CRV}$ and $\Phi^A$ as:

$$\zeta_{t-1}^{\Phi} = \text{get\_D}([\Phi_{t-1}, \Phi_{t-1}^{3CRV} \times P_{3CRV}], \Phi^A)$$

Beanstalk calculates $\Phi_{t-1}^{*}$ from $\zeta_{t-1}^{\Phi}$ as:

$$\Phi_{t-1}^{*} = \frac{\zeta_{t-1}^{\Phi}}{2}$$

Beanstalk calculates $\Delta b_{t-1}^{\Phi}$ for a given $\Phi_{t-1}^{*}$ and $\Phi_{t-1}$ as:

$$\Delta b_{t-1}^{\Phi} = \Phi_{t-1}^{*} - \Phi_{t-1}$$

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

$$\Delta B_{t-1} = \left\lfloor \Delta b_{t-1}^{\Phi} \right\rfloor$$
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.
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
- 0x90E00ACe148ca3b23Ac1bC8C240C2a7Dd9c2d7f5
- 0x8F942C20D02bE6c377D41445793068908E2250D0

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 0xb1bE0000C6B3C62749b5F0c92480146452D15423 as a sandbox for execution.

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

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

1. **Token Address:** The USDC token address is 0xA0b86991c6218b36c1d19D4a2e9E6b0cE3606eB48.
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

1. **Token Address:** The USDC token address is 0xA0b86991c6218b36c1d19D4a2e9E6b0cE3606eB48.
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

1. **Token Address:** The USDC token address is 0xA0b86991c6218b36c1d19D4a2e9E6b0cE3606eB48.
2. **Tokens to Raise:** The Fundraiser is for 250,000 USDC.
3. **Wallet to Send Tokens:** The tokens are sent to 0x21DE18B6A8f78eDe6D16C50A1676fB222DC08DF7 upon completion of the Fundraiser.

---

68 bean.money/bip-4
69 bean.money/bip-5
70 bean.money/bip-10

53
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, $q$ is the current block of $t$, $Ξ$ is the end of the current block, and $jadi$ is the current transaction; and
- Superscripts are modifiers.

Δ, ∂ and $f$ are ignored for the purposes of categorization and ordering in this glossary.

The following variables and terms are used throughout this paper:

14.11.1 Terms

*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;

*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;

*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* $\lambda$ for another, within the *Silo*;

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

*DAO* - Decentralized autonomous organization;

*DeFi* - Decentralized finance;

*Deposit* - An asset in the *Silo*;

*Deposit ID* - The concatenation of the $\lambda$ token address and the maximum *Grown Stalk* per BDV of $\lambda$ at the time of *Deposit*;
**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;

**Earned 🌸** - 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;

**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 Flood Whitelist;

**Flood Whitelist** - The whitelist of pools that Beanstalk sells Beans directly in when it mints extra Beans during a Flood;

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

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

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

**Harvest** - Redeem;

**Harvestable** - Redeemable;

**Harvested** - Redeemed;

**Humidity** - The interest rate on Fertilizer purchases;

**List** - Create an offer to sell;

**LP tokens** - Liquidity pool tokens;

**Market** - The Beanstalk-native DEX.;

**Maximum Soil** - The maximum Soil supply at a given block during a Season;

**Maximum Temperature** - The maximum Temperature Beanstalk is willing to offer during a Season;

**Minimum Soil** - The minimum Soil supply at a given block during a Season;

**Morning** - The first Q blocks of each Season;

**Mow** - Turn Grown Stalk into Stalk;
Oracle Whitelist - The whitelist of pools included in the calculation of $\Delta B_{t-1}$;

Order - Create an offer to buy;

Pause - Stop accepting gm function calls;

Paused - Beanstalk has stopped accepting gm function calls;

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

Plant - Turn Seeds associated with Earned $\$\$\$\$ into Seeds by Depositing the Earned $\$\$\$\$ 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 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;

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

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;

Season - Beanstalk-native discrete time;

Seed - A Beanstalk-native asset that Grow $1 \times 10^{-4}$ 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 $\$\$1$ 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;
Temperature - The interest rate on Bean loans;
Transfer - Send a Deposited asset;
TWA - Time weighted average;
Unfertilized Sprouts - Sprouts not yet Fertilized by Active Fertilizer;
Unharvestable Pods - Pods that are not yet redeemable;
Unpause - Resume accepting gm function calls;
Unpaused - Beanstalk has resumed accepting gm 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;
Voting Period - The period of time Stalkholders can vote on a BIP;
Withdraw - Remove from the Silo; and
Yield - Pods being created from Sown Ø.

14.11.2 Latin Alphabet Variables

\( A_{BIP} \) - 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 in Former Governance;
\( a_t \) - The award for successfully calling the gm function for \( t \);
\( B \) - The total Bean supply;
\( B^{BIP} \) - The total Beans minted via BIPs;
\( B_{t-1} \) - The Bean supply at the end of the previous Season;
\( \Delta B_{t-1} \) - The number of Beans that are minted and sold to return the price of \( Ø1 \) to its value peg;
\( \Delta B_{t-1}^\delta \) - The sum of liquidity and time weighted average shortages or excess of Beans across \( Ø:y \) liquidity pools on the Oracle Whitelist over the previous Season;
\( \Delta b_{t-1}^\Theta \) - The liquidity and time weighted average shortage or excess of Beans in the BEAN:ETH Well over the previous Season;
\( \Delta b_{t-1}^\Phi \) - The shortage of Beans in the BEAN:3CRV Curve pool at the end of the previous Season;
\( \Delta b_{t-1}^\Phi \) - The liquidity and time weighted average shortage or excess of Beans in the BEAN:3CRV Curve pool over the previous Season;
\( C_t \) - A Stalkholder’s total Seeds during \( t \);
\( C_\circ \) - A Stalkholder’s Seeds at the end of the block prior to the Exploit;
$C_\otimes$ - A Stalkholder’s Seeds upon Replant;
$c^\lambda$ - The number of Seeds per BDV of $\lambda$ Deposited;
$c^\lambda_t$ - The Seeds during $t$ for a given Deposit;
$D$ - The total number of Unharvestable Pods;
$\Delta D_t$ - The number of Pods that Ripen and become Harvestable at the beginning of each Season;
$D_\gamma$ - The number of Unharvestable Pods that grew prior to the Flood;
$d$ - The number of Pods that Yield from a given number of Sown $\ddot{o}$;
$E$ - 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 in Former Governance;
$E_{t}^{\text{min}}$ - The minimum timestamp Beanstalk accepts a $\varphi$ function call for a given $t$;
$E_{t}^{\text{first}}$ - The Ethereum timestamp of the first Sow in $t$ such that there is at most one Soil;
$\Delta E_t^\mu$ - The difference in time it took for the Beans to be Sown in all but at most one Soil over the previous two Seasons;
$\Delta E_t^{\text{first}}$ - The time of the first Sow such that Beans are Sown in all but at most one Soil in each Season;
$E_q$ - The timestamp a BIP was committed in Former Governance;
$E_{\Xi}$ - The current block timestamp;
$E_{\Xi}$ - The timestamp of the current block;
$E_{\Psi}$ - The timestamp Beanstalk last Unpaused;
$E_{\Sigma}$ - The timestamp of the last interaction with $\Sigma$;
$G_t$ - A Stalkholder’s total Grown Stalk that can be Mown during $t$;
$g^\lambda_t$ - The Grown Stalk from Seeds from $\lambda$ Deposits that can be Mown during $t$ to start earning Bean seigniorage for a given Deposit of a Stalkholder that last Mowed their Grown Stalk from $\lambda$ Deposits in $\varpi^\lambda$;
$h$ - The Temperature;
$h_{t}^{\text{max}}$ - The Maximum Temperature during $t$;
$h_{t_q}$ - The Temperature in block $q$ of $t$;
$K_{\text{min}}$ - The percentage of Stalk ownership necessary to submit a BIP;
$K_{\text{end}}$ - The percentage of Stalk ownership necessary for a submitter’s BIP to pass at the end of the Voting Period;
$K_t$ - A Stalkholder’s total Stalk during $t$;
$K_\otimes$ - A Stalkholder’s Stalk at the end of the block prior to the Exploit;
$K_\otimes$ - A Stalkholder’s Stalk upon Replant;
$k^\lambda$ - The number of Stalk per BDV of $\lambda$ Deposited;
$k^\lambda_t$ - The Stalk during $t$ for a given Deposit of a Stalkholder that last Mowed their Grown Stalk from $\lambda$ Deposits in $\varpi^\lambda$;
$L_\lambda^i$ - The total BDV of $Z_\lambda^i$ when Deposited;

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

$m_t$ - The number of Beans that Beanstalk mints at the beginning of each Season;

$N^\phi$ - The total Burnt $\phi$ over all Seasons;

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

$P_{t-1}$ - The inferred liquidity and time weighted average price of $\phi1$ compared to $V$ over the previous Season;

$P^\phi$ - The $\phi$ virtual price;

$P^\Omega$ - The $\Omega$ virtual price;

$P^\gamma$ - The $\gamma$ virtual price;

$Q$ - The length in blocks of the Morning;

$q$ - The current block of $t$;

$R^D$ - The Pod Rate;

$R_{t-1}^D$ - The Pod Rate at the end of the previous Season;

$S$ - Soil;

$S_{t\text{end}}$ - The Soil supply at the end of the Season;

$S_{t\text{start}}$ - The Soil supply at the beginning of the Season;

$\Delta S_t$ - The change in Soil from the beginning to the end of each Season;

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

$S_{tq}^{\text{max}}$ - The Maximum Soil in block $q$ of $t$;

$S_{tq}^{\text{min}}$ - The Minimum Soil in block $q$ of $t$;

$t$ - The current Season;

$U$ - The total Sown $\phi$ over all Seasons;

$u$ - The number of Sown $\phi$;

$u_t$ - The number of Sown $\phi$ during $t$;

$\frac{\partial u_t}{\partial t}$ - The rate of change of $u_t$ over the previous two Seasons;

$V$ - The value peg for $\phi1$;

$x$ - 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 $y$;

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

$Z_\lambda^i$ - The total number of $\lambda$ Deposited during Season $i$. 

59
14.11.3 Mathfrak Style Latin Alphabet Variables

A - Active Fertilizer;
D - The total Unfertilized Sprouts;
\Delta D - The total Sprouts Fertilized by Fertilizer;
\Delta D_f - The number of Unfertilized Sprouts that are Fertilized by Active Fertilizer and become Rinsable at the beginning of each Season;
\phi - The number of Sprouts ultimately Fertilized by Available Fertilizer purchased with Humidity \( \phi \);
\phi - The total Fertilizer;
\phi - The Humidity;
\Phi - The percentage of Ripe assets received for Chopping a pro-rata portion of Unripe assets;
\Phi - The number of \( \Phi \) received for Chopping a given \( \phi \);
\Phi - The number of Beans received for Chopping a given \( \phi \);
\Phi - Ripe \( \Phi \);
\Delta \Phi - The change in Ripe \( \Phi \) for a given \( \Delta D \);
\Phi - Ripe \( \Phi \);
\Delta \Phi - The change in Ripe \( \Phi \) for a given purchase of Fertilizer;
\Phi - The Ripe \( \Phi \) prior to a purchase of Fertilizer;
\phi - The total Fertilizer sold;
\phi - The Fertilizer sold prior to a purchase of Fertilizer;
\Delta \phi - A purchase of Fertilizer;
\Phi - Used Fertilizer;
\Phi - Available Fertilizer;
\Phi - Available Fertilizer purchased with Humidity \( \phi \);
\chi - The percentage of Fertilizer sold;
\Delta \chi - The change in \( \chi \) between (1) \( \varphi \) or (2) the Replant, if \( \varphi = 0 \), and \( t \);
\chi - The percentage of Fertilizer sold prior to Replant;
\phi - The current total Unripe \( \Phi \);
\phi - The total Unripe \( \Phi \);
\phi - The total Unripe \( \Phi \) at the time of Replant;
\phi - Unripe \( \Phi \);
\phi - The minimum number of Unripe \( \Phi \) received for Converting Deposited Unripe Beans within the Silo;
\phi - Unripe \( \Phi \); and
\phi - The minimum number of Unripe Beans received for Converting Deposited Unripe \( \Phi \) within the Silo.
14.11.4 Greek Alphabet Variables

ζΦ - The price invariant for the BEAN:3CRV Curve pool;
ζΦ_{t-1} - The flash-loan-resistant price invariant for the BEAN:3CRV Curve pool;
ζΦ_{t-1} - The price invariant for Φ at the end of the previous Season;
ζΦ_{t-1} - The liquidity and time weighted average price invariant for Φ over the previous Season;
ζΩ - The flash-loan-resistant price invariant for the LUSD:3CRV Curve pool;
ζℸ - The flash-loan-resistant price invariant for the old BEAN:3CRV Curve pool;
η - The last Season a Stalkholder called the plant function;
ηc - The Plantable Seeds associated with a Stalkholder’s η that can be Planted to start earning Grown Stalk;
ηcJ - A Stalkholder’s Plantable Seeds at the end of the block prior to the Exploit;
Θ - BEAN:ETH Well LP tokens;
∂Θ∂ - The inter-block MEV manipulation resistant derivative of the Θ LP token supply with respect to Beans;
ΘEMA,0 - The inter-block MEV manipulation resistant instantaneous Bean reserves in the Multi Flow Pump of the BEAN:ETH Well in the current transaction;
ΘEMA_{ETH,0} - The inter-block MEV manipulation resistant instantaneous ETH reserves in the Multi Flow Pump of the BEAN:ETH Well in the current transaction;
ΘETH - The number of ETH in the BEAN:ETH Well’s Reserves in the current transaction;
Θmin - The minimum number of Θ received for Converting to Deposited Θ within the Silo;
Θmin(λ) - The minimum number of Beans that must be in the BEAN:ETH Well in order for the oracle to be considered;
ΘSMA,0 - The inter-block MEV manipulation resistant TWA Bean reserves in the Multi Flow Pump of the BEAN:ETH Well from the beginning of the Season to the current transaction;
ΘSMA_{ETH,0} - The inter-block MEV manipulation resistant TWA ETH reserves in the Multi Flow Pump of the BEAN:ETH Well from the beginning of the Season to the current transaction;
Θ* - The data associated with the Well Function of Θ;
Θ5 - The number of Beans in the BEAN:ETH Well’s Reserves in the current transaction;
Θ\^{\ast}t - The optimal liquidity and time weighted average number of Beans in Θ over the previous Season;
Θ\rightarrow - The number of Θ LP tokens Converted;
Λ - The Deposit Whitelist;
λ - Θ and other assets on the Deposit Whitelist;
fλ(λ) - The function to calculate the flash-loan-resistant Bean-denominated-value for a given number of λ Deposited;
fλ→λ’(λ) - The function to determine the number of λ’ received for Converting a given number of λ;
σ - A control variable used to calculate the Temperature during the Morning;

Φ - BEAN:3CRV LP tokens;

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

Φ^{min} - The minimum number of Φ received for Converting to Deposited Φ within the Silo;

Φ^{A} - The A parameter of Φ;

Φ^{β}_{t-1} - The number of Beans in Φ at the end of the previous Season;

Φ^{ρ}_{t-1} - The optimal number of Beans in Φ at the end of the previous Season;

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

Φ^{Ξ}_{t-1} - The flash-loan-resistant total number of Φ;

Φ^{π}_{t-1} - The number of Beans in the BEAN:3CRV Curve pool at the end of the last block;

Φ^{3CRV}_{Ξ-1} - The number of 3CRV in the BEAN:3CRV Curve pool at the end of the last block;

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

Φ^{ρ}_{t-1} - The liquidity and time weighted average number of Beans in Φ over the previous Season;

Φ^{3CRV}_{t-1} - The time weighted average number of 3CRV in Φ over the previous Season;

Ω - The LUSD:3CRV Curve pool;

Ω^{A} - The A parameter of Ω;

Ω^{Ξ}_{t-1} - The flash-loan-resistant total number of Ω;

Ω^{3CRV}_{Ξ-1} - The number of 3CRV in the LUSD:3CRV Curve pool at the end of the last block; and

Ω^{LUSD}_{Ξ-1} - The number of LUSD in the LUSD:3CRV Curve pool at the end of the last block.

14.11.5 Glyph Variant Greek Alphabet Variables

ϑ - An approximation of the current price of ETH in Beans;

κ^{λ} - The last Season a Stalkholder Mowed their Grown Stalk from λ Deposits;

ϖ^{Ξ} - An approximation of the gas fee of the current block denominated in Wei;

ϱ - An approximation of the gas used to execute the gm function call;

ς - The difference between gasleft at the beginning and end of the gm function call;

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

φ^{C}_{t} - The number of Revitalized Seeds that can be Enrooted by a Stalkholder during t; and

φ^{K}_{t} - The number of Revitalized Stalk that can be Enrooted by a Stalkholder during t.
14.11.6 Hebrew Alphabet Variables

ℶ - Old BEAN:ETH Uniswap V2 LP tokens;
™ - The current total number of ℶ in the current block;
ℶ^ - The last traded price was a function of the current number of Beans in ℶ;
ℶ^ - The time weighted average number of Beans in ℶ from the start of the current Season to the current block;
ℷ - Old BEAN:LUSD Curve LP tokens;
ランキング - Old BEAN:3CRV Curve LP tokens;
やっぱ - The A parameter of ℹ;
ℸא - The flash-loan-resistant total number of ℹ;
ℸא - The number of Beans in the old BEAN:3CRV Curve pool at the end of the last block; and
ℸא - The number of 3CRV in the old BEAN:3CRV Curve pool at the end of the last block.

14.11.7 Symbol Variables

וף - Bean;
וף - The minimum number of Beans received for Converting to Deposited Beans within the Silo;
וף - A new liquidity pool that consists of Beans and y;
$ - 1 US Dollar;
$ETH - The inter-block MEV manipulation resistant USD price of 1 ETH;
$ETH(ν) - The USD price of 1 ETH in the ETH:USDC 0.05% fee Uniswap V3 pool;
$ETH(τ) - The USD price of 1 ETH in the ETH:USDT 0.05% fee Uniswap V3 pool;
$ETH(χ) - The USD price of 1 ETH from the ETH/USD Chainlink data feed;
$^[ETH(ν/χ)] - The percent difference between $ETH(χ) and $ETH(ν);
$^[ETH(τ/χ)] - The percent difference between $ETH(χ) and $ETH(τ);
$LUSD(Ω) - The USD price of 1 LUSD from the LUSD:3CRV Curve pool;
$LUSD(Φ) - The USD price of 1 in the BEAN:ETH Well;
$LUSD(Φ) - The USD price of 1 in the BEAN:3CRV Curve pool;
$LUSD(Φ) - The flash-loan-resistant USD price of 1 in the BEAN:3CRV Curve pool; and
$LUSD(Φ) - The USD price of 1 in the old BEAN:3CRV Curve pool; and
$ - An approximation of the cost to call the gm function in Beans in the current 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 $P_0 = 1$ 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.\(^7\)
  - 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.

\(^7\) 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 \( f^2 \) 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 \( \Lambda_h \) to \( b_\Omega, \Omega \) and \( \Lambda_\Omega \), 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.\(^{72}\)
- 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.\(^{73}\)
- 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.

\(^{72}\) bean.money/bip-6
\(^{73}\) bean.money/bip-7
– Changed variable $E_f$ to $E_\Psi$ 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 $S_{\text{start}}^t$ 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 $K_{\text{min}}$, $Ã^{\text{Silo}}$, and $\xi$ to the Glossary.
– Changed $S_{\text{end}}^{t-1}$ to $S_{\text{end}}^t$ in the Glossary for consistency.
– Removed $B_t$, $S_{\text{max}}^t$, and $R_S^{\text{max}}$ 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-12.74
  – 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 \( c_{\lambda}, c_{\lambda t}, K_{\lambda t} \), \( l, \lambda, z_{\lambda} \) and \( Z_{\lambda} \) to the Glossary.
  – Changed \( G \) to \( \mu \), \( \Lambda \) to \( \phi \) and \( \Lambda_{\text{Silo}} \) to \( \phi_{\text{Silo}} \) in the Glossary.
  – Removed \( b_{\Omega}, c_{t}, c_{\Lambda t}, k_{t}, k_{\Lambda t}, l_{\Lambda i}, \lambda, \omega, \lambda_{\omega}, z_{i}, z_{\Lambda i}, z_{\Lambda}, z_{\Lambda}: \) and \( \Omega \) 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.
  – 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 \( a^{\text{BIP}} \) and \( A^{\text{BIP}} \).
  – Corrected Section 6.5.5 to reflect the correct rate and duration that \( a^{\phi} \) compounds.
  – Updated the equation for \( B \) in Section 8.4.5 to include \( B^{\text{BIP}} \).
  – Corrected Section 8.4.8 to reflect the Weather changes when \( R^D \) equals \( R^{D_{\text{lower}}} \), \( R^{D_{\text{upper}}} \) 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 \( B^{\text{BIP}}, BDV, c^\phi, c^\Phi, E_{\Xi}, E_{\phi}, E_{\Psi}, \zeta^\phi, g^\phi(z^\phi), g^\Phi(z^\Phi), g^{3\text{CRV}}(z^{3\text{CRV}}), k^\phi, k^\Phi, P^\phi, P^{3\text{CRV}}, \Phi, \phi_{A^\phi}, \phi_\Xi, \phi_{E-1}, \Phi_{E-1}, \Phi^{3\text{CRV}}_{E-1} \) and \( \Phi^{3\text{CRV}}_{E-1} \) 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.

\[\text{bean.money/bip-12}\]
\[\text{bean.money/bip-13}\]

67
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.
- Added $\Delta E_t^{\text{first}}$ and $E_t^{\text{first}}$ to the Glossary.
- Changed the definition of $\Delta E_t^{\text{in}}$ in the Glossary.
- Removed $\Delta E_t^{\text{last}}, \Delta R_{t-1}^S, \frac{\partial R_{t-1}^{\text{upper}}}{\partial t}, E_t^{\text{first}}, R^S, R^S_{\text{min}}, R^S_{\text{mid}}$ and $R^S_{\text{start}}$ 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 $\alpha$ to the Silo Whitelist, as amended by BIP-16.
- Added $\Omega(\Phi), LUSD(\Omega), \alpha, c^{\alpha}, \zeta^\Omega, g^{\alpha}(z^{\alpha}), k^{\alpha}, P^\alpha, P^\Omega, \Phi^A, \Omega^A, \Omega_{-1}, \Omega_{-1}^{\text{LUSD}}$ and $\Omega_{-1}^{\text{SCRV}}$ 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 $P^\Phi$ and $P^{\text{SCRV}}$ 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 $\Delta b_{t-1}^\Phi$, as amended by EBIP-2.
- Updated BIP links throughout the whitepaper to the Beanstalk Governance Proposals GitHub Repository.
- 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.
- 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.

---

76 [bean.money/bip-15]
77 [bean.money/bip-16]
78 [bean.money/bip-2]
79 [github.com/BeanstalkFarms/Beanstalk-Governance-Proposals]
80 [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-29.81,82
  – 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 $K^{\text{min}}$ to the Glossary.
  – Moved $\Phi$ in the Glossary.
  – Corrected a typo in the definition of $\Delta R^2$ 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-30.83
  – 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.84
  – 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.

81 bean.money/ebip-3
82 bean.money/bip-29
83 bean.money/bip-30
84 github.com/BeanstalkFarms/Beanstalk-Whitepaper
2.4.0 (July 3, 2023)

- Changed the `sunrise` function to the `gm` function throughout the whitepaper, as amended by BIP-34.85
- Updated the second paragraph of Section 4 to reflect that blocks on Ethereum are no longer mined after the Merge.
- Updated Section 4 to reflect the changes to the award for successfully calling the `gm` function, as amended by BIP-34.
- Changed $E_t$ to $E_Ξ$ throughout the whitepaper for clarity.
- Corrected a typo in the second paragraph of Section 5.5.1.
- Updated Section 5.5.1 to reflect that (1) a Stalkholder’s vote is a function of their Stalk at the beginning of the Voting Period that still exists, (2) Stalkholders have the ability to delegate their vote to any other user, and (3) the submitter of a BIP must own more than $K_{end}^{min}$ percent of total Stalk at the end of the Voting Period in order for the BIP to be able to pass, as amended by BIP-35.86
- Updated Section 5.5.2 to (1) improve clarity around when the Voting Period begins, (2) account for the BIP passing or failing based on the total outstanding Stalk at the beginning of the Voting Period that still exists, and (3) reintroduce the 24 hour period after the beginning and before the end of the Voting Period where a BIP cannot be passed via supermajority, as amended by BIP-35.
- Updated the BIP inputs in the second paragraph of Section 5.5.4 to reflect the current state of off-chain governance.
- Updated Sections 6, 6.1 and 6.3 to reflect the changes to the Soil supply and Temperature introduced by the Morning, as amended by BIP-34.
- Updated Section 8.7 to reflect that (1) demand for Soil is now based on the number of Sown $δ$ each Season rather than the change in Soil at the beginning and end of each Season and (2) Beanstalk considers demand for Soil increasing if $\Delta E^u_{t-1} < 600$, as amended by BIP-34.
- Updated the Risk section to include a citation for the Beanstalk DAO Disclosures.
- Replaced the Beanstalk audit report citation links in the Risk section with the link to the Beanstalk Audits GitHub Repository.88
- Changed $\frac{\partial \Delta S}{\partial t}$ to $\frac{\partial u}{\partial t}$ in Future Work for consistency.
- Updated the notation of percentage values in the Current Parameters for clarity.
- Corrected $w_1$ to $h_1$ in Current Parameters.
- Changed $\frac{\partial \Delta S}{\partial t}$ lower and $\frac{\partial \Delta S}{\partial t}$ upper to $\frac{\partial u}{\partial t}$ lower and $\frac{\partial u}{\partial t}$ upper, respectively, in Current Parameters for consistency.
- Added $K_{end}^{min}$, $Q$ and $\sigma$ to Current Parameters.
- Removed $R_{S}^{min}$, $R_{S}^{max}$, and $\frac{\partial R}{\partial t}$ upper from Current Parameters.
- Updated Section 14.2.2 to reflect that the definitions of $S_{\Xi}^{(\Phi)}$, $\Phi_{\Xi}^{\phi}$, $\Phi_{3CRV}^{\phi}$, $P_{3CRV}$, $\Phi^{A}$, $P^{\Phi}$, $\xi_{\Xi}^{\phi}$, and $\Phi_{\Xi}^{\phi}$ were moved to Section 4.
- Updated the intro of the Glossary to include that the $q$ subscript is the current block of $t$.85 bean.money/bip-34 86 bean.money/bip-35 87 bean.money/disclosures 88 github.com/BeanstalkFarms/Beanstalk-Audits
- Added $h_{\text{max}}$, $h_{\text{tq}}$, $K_{\text{end}}$, Maximum Soil, Minimum Soil, Maximum Temperature, $w_{\text{Z}}$, $Q$, $q$, $\theta$, $S_{\text{max}}$, $S_{\text{min}}$, $\sigma$, $\sigma$, $u$, and $\frac{du}{dt}$ to the Glossary.
- Updated the definitions of $E_{\text{Z}}$, $S_{\text{end}}^{t}$ and $S_{\text{start}}^{t}$ in the Glossary.
- Removed $E_{\text{start}}^{\lambda}$ and $S_{\text{min}}$ from the Glossary.
- Added language to $a^{\delta}$, $E_{\text{q}}$ and $E_{\text{BIP}}$ in the Glossary to clarify that they refer to Former Governance.
- Corrected a typo under Version 2.0.1 in the Whitepaper Version History.

- **2.5.0 (July 31, 2023)**
  - Added Beanstalk Farms as an author to the whitepaper.
  - Updated Sections 5.1 and 5.4 to reflect that Grown Stalk must now be Mown for each $\lambda$, as amended by BIP-36.\(^{89}\)
  - Updated Section 5.3 to reflect the implementation of Deposits as ERC-1155 Standard tokens, as amended by BIP-36.
  - Updated Figure 1, Sections 5.3, 8.13, 11.6, 14.8.1.4, and Future Work to reflect the removal of the Withdrawal Freeze and Oversaturated, as amended by BIP-36.
  - Updated Section 5.4 to reflect (1) that Beans minted to the Silo are distributed to Stalkholders and become Earned 2 10 blocks past the beginning of the Season in which they were minted, (2) the changes to the Deposit accounting system and (3) that Stalk is no longer lost due to rounding during Conversions, as amended by BIP-36.
  - Updated Section 5.5.5 and Future Work to reflect the expectation that permissionless governance need not be reimplemented and that future BIPs will remove governance entirely.
  - Added the ERC-1155 Standard to the Risk section.
  - Removed $\xi$ from Current Parameters.
  - Corrected the formatting of Stalk and Seeds in Sections 14.2 and 14.5.
  - Updated Sections 14.2.3 and 14.2.4 to reflect that the Seeds per BDV for $\gamma$ and $\delta$ are 0, as amended by BIP-36.
  - Added Deposit ID to the Glossary.
  - Updated the definitions of $D_{\gamma}$, $g_{\lambda}$ and $k_{\lambda}$ in the Glossary.
  - Changed $\varpi$ to $\varpi$ in the Glossary.
  - Removed Frozen, Oversaturated, Withdrawal Freeze and Withdrawn assets from the Glossary.
  - Corrected the formatting of Minimum Soil and $S_{\text{min}}$ in the Glossary.

- **2.6.0 (October 16, 2023)**
  - Updated the email address for Beanstalk Farms.
  - Updated Section 4 to reflect that the current price of ETH in Beans is calculated using the Multi Flow Pump on the BEAN:ETH Well, as amended by BIP-37.\(^{90}\)
  - Updated the citation for BIP-20\(^{91}\) in Section 7.3.3.
  - Corrected a typo in the second paragraph of Section 8.
  - Corrected typos in the second and seventh paragraphs of Section 8.2.
  - Updated language in the third paragraph of Section 8.2 for accuracy.
  - Updated the citation for WETH in Section 8.2.

\(^{89}\) bean.money/bip-36
\(^{90}\) bean.money/bip-37
\(^{91}\) bean.money/bip-20
– Updated Section 8.2 to reflect the instances in which Beanstalk uses a Chainlink data feed as part of the price oracle, as amended by BIP-37.
– Corrected a typo in the first paragraph of Section 8.13.
– Updated Section 8.13 to reflect that (1) at the beginning of each Season during a Flood, Beanstalk returns the price of $\hat{p}$1 in each liquidity pool on the Flood Whitelist to its value peg by minting additional Beans and selling them directly in the pools, and (2) liquidity pools can be added to and removed from the Flood Whitelist via Beanstalk governance.
– Overhauled the Risk section to reflect the current state of Beanstalk.
– Removed support of additional Pipelines from the Future Work section.
– Added $\Theta^{\min(\hat{p})}$ to the Current Parameters.
– Reordered the Current Parameters to be consistent with the updated Glossary.
– Updated Sections 14.2.1 and 14.2.2 to reflect the change in Seeds per BDV for $\hat{p}$ and $\Phi$ to 3 and 3.25, respectively, as amended by BIP-37.
– Updated Section 14.2.2 to include the definitions of $\mathcal{S}_{\mathcal{E}-1}$, $\Phi_{\mathcal{E}-1}$, $\Phi_{3\text{CRV}}$, $P_{3\text{CRV}}$, $\Phi_A$, $P$, $\zeta_{\mathcal{E}-1}$, and $\Phi_{\mathcal{E}-1}$, which were removed from Section 4.
– Fixed typos in the BDV functions for Sections 14.2.3 and 14.2.4.
– Added Section 14.2.5 to reflect the addition of $\Theta$ to the Deposit Whitelist, as amended by BIP-37.
– Fixed a typo in the fourth paragraph of Section 14.4.2 and under Conditions in Sections 14.4.3, 14.4.4 and 14.4.5.
– Updated the Convert functions in Sections 14.4.2, 14.4.3, 14.4.4 and 14.4.5 for accuracy.
– Added Sections 14.4.6 and Sections 14.4.7 to reflect the addition of $\hat{p} \rightarrow \Theta$ and $\Theta \rightarrow \hat{p}$ Conversions to the Convert Whitelist, as amended by BIP-37.
– Amended Sections 14.5.2 and 14.5.3 for clarity.
– Fixed a typo under Seeds per BDV in Section 14.5.3.
– Amended the intro to Section 14.6 for clarity.
– Fixed a typo and defined $B_{t-1}$ in Section 14.6.1.
– Added Section 14.6.2 to reflect the addition of $\Theta$ to the Oracle Whitelist, as amended by BIP-37.
– Renamed Section 14.7 to Flood Whitelist and updated the intro for clarity.
– Updated Section 14.9.2 to reflect the new Pipeline contract address used by Depot, as amended by BIP-37.
– Overhauled the ordering of the Glossary for readability.
– Added Flood Whitelist, TWA, $B_{t-1}$, $\Delta \Theta_{t-1}$, $m_t$, $\Theta_{\text{EMA}}$, $\Theta_{\text{ETH},t}$, $\Theta_{\text{ETH},0}$, $\Theta_{\text{ETH},\tau}$, $\Theta^{\min}$, $\Theta^{\min(\hat{p})}$, $\Theta_{\text{SMA},t}$, $\Theta_{\text{SMA},0}$, $\Theta_{\text{*,t}}$, $\Theta_{\text{*,0}}$, $\Theta_{\text{*,\tau}}$, $\Theta \rightarrow$, $\Theta_{\text{ETH}}$, $\Theta_{\text{ETH}(\nu)}$, $\Theta_{\text{ETH}(\tau)}$, $\Theta_{\text{ETH}(\chi)}$, $\Delta \Theta_{\text{ETH}(\nu/\chi)}$, $\Delta \Theta_{\text{ETH}(\tau/\chi)}$ and $\Theta_{\text{*,t}}$ to the Glossary.
– Updated the definitions for Flood, $B^{\text{BIP}}$, $\Phi^{\min}$ and $\hat{p}^{\min}$ in the Glossary.
– Removed $\kappa$ from the Glossary.