Intro

HEX is a project to recreate a common banking product called a Time Deposit. It is an ERC-20 token and fully automated in the form of a smart contract on the Ethereum blockchain. Information and a FAQ is available at https://HEX.com.

The purpose of this document is to walk through the features and bullet points of the project and verify or explain them in terms of what the smart contract actually says. The project founder, Richard Heart, has provided in-development versions of the code to the community via the Telegram channel http://t.me/HEXcrypto. I have participated in the discussion in that channel, reviewed the code, and even pointed out a bug or two (which were fixed in the next iteration). I do not wish to provide advice or editorial comment on any of the features. This is meant to be an objective explainer of what the contract will do.

For a more in depth discussion of Staking and the gains to be made, see this companion document https://bit.ly/hex-staking-guide.

Comments disabled because a guy was using the functionality to vandalize the doc.

Pre-Launch

The HEX folks will create a snapshot of the state of the BTC blockchain at some future block height. This means whatever addresses you control will have their balance in BTC recorded and referenced by the contract. **This part is not in the smart contract but it is referenced from the smart contract**

Before launch, 3 things will be made available to the public for confirmation of accuracy.
1. The snapshot UTXO set before editing (see GoxMeNot)
2. The snapshot UTXO set after editing
3. The code for generating the merkle tree and top hash

With these, any interested party would be able to confirm
1. The snapshot was taken properly
2. It was edited properly
3. The top hash in the contract is correct

There will be a claim tool on the website that assists the user in signing the target ETH address with your BTC private key. Instructions will be provided when the tool is available. **This part is not in the contract but the contract is provided the signed statement material, BTC public key, and Ethereum address which are verified by the contract**

Pre-Claim

There will be one day of the contract being live that supports only a Transform Lobby. This ensures that folks who buy HEX via the transform process may begin staking at the same time as BTC claimers. Details on this process are below in The Transform Lobbies. One notable difference about the first day is that 1,000,000,000 HEX are distributed on the first day. This is congruent with the narrative that lobbies are filled with unclaimed coins and on the first day before claiming, there are no unclaimed coins.

Basic Flow

The contract issues 10,000 HEX per BTC held by the BTC address at the time of the snapshot. We have been promised a 2 week warning for the snapshot. For example, if we are given

warning on April 20th, the snapshot should occur on May 3rd with the contract going live on May 4th and the first claim day being May 5th. It doesn't matter how much bitcoin I have at my address(es) on April 20th. It matters only how much bitcoin is confirmed at my address(es) on May 3rd. Suppose I get the warning and move 3 BTC to address A, and 2 BTC to address B on May 2nd. After the snapshot on May 4th I can then do anything I want with those bitcoins including sell them. I sell them on May 6th. Then on May 19th (first claim day + 2 weeks) I go to claim I will generate 2 signed statements, one from address A and one from address B. Note that each wallet can contain multiple funded addresses. You need to make a separate claim for each funded address. I decide to consolidate on a single ETH address for both claims. The contract validates my signature and credits me with **10,000 * 3 + bonuses - penalties** (described shortly) HEX for address A to my ETH address because on May 3rd I had 3 BTC in that wallet. The contract validates my signature and credits me with **10,000 * 2 + bonuses - penalties** HEX for address B to my ETH address for the same reason. I will also receive some bonuses and penalties which will be described momentarily.

For BTC claims, the contract immediately stakes 90% of the claimed HEX for a minimum of 350 days. This will be mediated by the UI and allow for longer stakes (the #days is passed as a claim parameter). The remaining 10% is minted to the claimant address. In the above example, 90% of the HEX (i.e. **45000 + bonuses - penalties**) is staked for a user-provided length of no fewer than 350 days. The remaining **5000 + bonuses - penalties** is minted and available immediately to stake for any length of time or market sell.

Now I have some HEX. I can transfer it like any other ERC-20 token or use a contract function to Stake it. Staking means I lock up my HEX in the contract, reducing my transferable balance. The benefit of Staking is that when I end my Stake period, the contract will return not only the HEX I put in, but additional HEX based on an interest rate and additional bonus payout features coded into the contract.

Claim Examples

The claim bonus is better and the penalties are lighter the closer to launch date you stake. The bonuses are described in detail below so for these examples they will just be applied. I will assume a 5 BTC claim for these examples

Normal Claims

Launch Day

5 BTC * 10,000 HEX/BTC * 1 (*no late penalty*) * 1.2 (*speed bonus*) = **60,000 HEX**

Two Weeks Post Launch

5 BTC * 10,000 HEX/BTC * 0.96 (*late penalty*) * 1.192 (*speed bonus*) = **57,216 HEX**

One Month Post Launch

5 BTC * 10,000 HEX/BTC * 0.914 (*late penalty*) * 1.1829 (*speed bonus*) = **54,073 HEX**

Six Months Post Launch

5 BTC * 10,000 HEX/BTC * 0.5 (*late penalty*) * 1.1 (*speed bonus*) = **27,500 HEX**

Last Eligible Claim Day (~1 year post launch)

5 BTC * 10,000 HEX/BTC * 0.0029 (*late penalty*) * 1.00057 (*speed bonus*) = **143 HEX**

Whale Claims (1,000 BTC <= claim < 10,000 BTC)

Launch Day

5000 BTC * 10,000 HEX/BTC * 0.3889 (*silly whale*) * 1 (*no late penalty*) * 1.2 (*speed bonus*) = **23,333,333 HEX**

Two Weeks Post Launch

5000 BTC * 10,000 HEX/BTC * 0.3889 (*silly whale*) * 0.96 (*late penalty*) * 1.192 (*speed bonus*) = **22,250,667 HEX**

One Month Post Launch

5000 BTC * 10,000 HEX/BTC * 0.3889 (*silly whale*) * 0.914 (*late penalty*) * 1.1829 (*speed bonus*) = **21,028,571 HEX**

Six Months Post Launch

5000 BTC * 10,000 HEX/BTC * 0.3889 (*silly whale*) * 0.5 (*late penalty*) * 1.1 (*speed bonus*) = **10,694,444 HEX**

Last Eligible Claim Day (~1 year post launch)

5000 BTC * 10,000 HEX/BTC * 0.3889 (*silly whale*) * 0.0029 (*late penalty*) * 1.00057 (*speed bonus*) = **55,587 HEX**

Mega Whale Claims (>= 10,000 BTC)

Launch Day

50,000 BTC * 10,000 HEX/BTC * 0.25 (*silly whale*) * 1 (*no late penalty*) * 1.2 (*speed bonus*) = **150,000,000 HEX**

Two Weeks Post Launch

50,000 BTC * 10,000 HEX/BTC * 0.25 (*silly whale*) * 0.96 (*late penalty*) * 1.192 (*speed bonus*) = **143,040,000 HEX**

One Month Post Launch

50,000 BTC * 10,000 HEX/BTC * 0.25 (*silly whale*) * .914 (*late penalty*) * 1.1829 (*speed bonus*) = **135,183,673 HEX**

Six Months Post Launch

50,000 BTC * 10,000 HEX/BTC * 0.25 (*silly whale*) * 0.5 (*no late penalty*) * 1.1 (*speed bonus*) = **68,750,000 HEX**

Last Eligible Claim Day (~1 year post launch)

50,000 BTC * 10,000 HEX/BTC * 0.25 (*silly whale*) * 0.0029 (*no late penalty*) * 1.00057 (*speed bonus*) = **357,347 HEX**

The Transform Lobbies

The other way to acquire HEX is the enter a Transform Lobby (called the Adoption Amplifier on the website). Every day, 1/350th of the unclaimed BTC is converted to HEX. ETH accumulated in the prior day is converted into HEX. For example, if there are 350,000 unclaimed bitcoins at the end of day 200, one thousand of them will be converted to HEX, i.e. 10,000,000 HEX (10,000 * 1000). If 500 ETH were accumulated over the course of day 200, any time on day 201 or after a user could claim HEX equal to 10,000,000 * <ETH they contributed> / total ETH. If they sent 1 ETH that day, they would collect 20,000 HEX.

Entering a transform lobby entails sending ETH to the contract function *joinXfLobby*. All ETH sent during a given day is tallied and any following day, *leaveXfLobby* may be called to claim your share of the transformed HEX. It's possible to make multiple entries in a day and the *leaveXfLobby* function takes both the day# and number of entries to resolve as inputs.

Note: The Transform system also pays for referrals so if you're using or promoting a referral link, this is another avenue to earn HEX.

Staking

Staking is invoking a function on the smart contract to commit your HEX for a time period. Currently the unit of time is days. I may Stake 10,000 HEX for 10 days, for example. During this time I may not access those HEX, but at the end of 10 days I end my Stake and receive my 10,000 + payouts. This is the Certificate of Deposit functionality.

Payouts are drawn from a pool based on my share of total Shares. I use “Shares” here intentionally because staking bonuses are calculated as a scaling of your HEX into Shares. Meaning, if I stake 10 HEX and qualify for 40% bonuses, then my 10 HEX become 14 Shares. My payout is based on my Shares divided by the total number of Shares, not HEX divided by the total number of HEX. This is important because it means that accruing bonuses is how to get the best payout, perhaps even more economical than starting with more HEX. It’s much easier to get a 40% bonus than to buy 40% more bitcoins, see LongerPaysBetter/BiggerPaysBetter below.

Share Price

In order to ensure that longer and larger stakes pay better over time, there is a pricing mechanism built into the contract. Every time a stake is ended the gains for that stake are calculated in the form of a share price which all future stakers will pay to convert their HEX into shares. An important note here is that HEX’s base unit is the Heart. Hearts are to HEX as Satoshis are to Bitcoin. There are 100,000,000 Hearts per HEX.

The share price at launch will be 1 share per Heart. The way the price moves is related to return on investment for a stake. For example, if on day 5 someone ends their stake and has a 20% gain, it translates into a share price of ~1.2 Hearts per share. If that user wishes to stake again, their Hearts will be divided by 1.2 to determine their number of shares. Since Ethereum only supports integer math, this is accomplished using a scalar which is listed below.

The exact formula is:
(BPB + cappedHearts) * stakeReturn * SHARE_RATE_SCALE / ((LPB * shares) / (LPB + extraStakedDays) * BPB);
Defining those terms:

- “BPB” is 10 x maximum value for BiggerPaysBetter bonus
- “stakeReturn” is the total amount paid out for ending the stake
- “SHARE_RATE_SCALE” is a scalar to maintain precision in the output (at time of writing it’s 5 decimal places)
- “shares” is the number of shares in that stake
- “LPB” is 1820
- “extraStakedDays” is the lesser of 3640 and (stakedDays - 1)
- “CappedHearts” is the lesser of your stakeReturn and the maximum BiggerPaysBetter figure

This means that it’s a calculation of your gains, adjusted for the additional BiggerPaysBetter bonus you will get for staking your returned HEX.

The intent is that the math ought to work out such that you should only ever be able to re-enter a stake with at best the same number of shares you just cashed in.

In this way, it guarantees that longer, bigger stakes will pay better over time because nobody else can play games to get more shares in the system. This also means it's always better to stake now because the price will only ever go up.

Payout Calculation and Interest

The contract accumulates a payout pool per day. The payout pool is filled with the daily inflation of 0.009955% of the total coin supply. This comes out to 3.69% per 52 weeks, compounded daily. There are additional inputs to the payout pool discussed below. When ending your stake at the end of your commitment, the contract goes through each day of your term and accumulates a total payout from your Shares/Total Shares * payout for the day. The contract mints the HEX and credits you with them.

Unstaking Gotchas

Early/Emergency Unstake

The contract has a feature whereby a user can terminate their stake before the committed time. The user pays a penalty for this. The contract calculates the payout for ½ (rounded up) the days committed, e.g. 182 for a 52 week commitment, and subtracts that from the funds returned to the user. **It will always apply at least a 90 day penalty**. For example, I stake for 52 weeks (about 1 year) and emergency unstake after 266 days (38 weeks). The contract calculates my payout for days 1 - 182 and flags that as the penalty. The funds returned to me are then my Principal + (Days 183 to 266) payouts.

If the user emergency unstakes before ½ their days are served or if their stake is fewer than 179 days due to the 90 penalty day minimum, the penalty can cut into Principal. The contract just calculates ½ the days or 90, whichever is larger, of payouts, estimating for future days unserved, and subtracts that from the funds returned.

For example, I stake for 364 days and then unstake at day 140. Several steps take place to determine my final returned coins.

- The contract calculates my payout for 140 days then estimates a penalty
- Because I have not served half my commited days it estimates the difference by applying a ratio of **half-committed-days / days-served * payout**
- In this case that is 182 days (half of 364 committed days) divided by 140 days served. The penalty is therefore **182/140 * payout** (½ days divided by served days).
- My returned HEX is equal to **Principal + payout - penalty**.
    - The penalty can be defined as a scaled version of the payout so my net HEX returned will be **Principal + (140/140 * payout) - (182/140 * payout)**
    - Which simplifies to **Principal - (42/140 * payout)**.

- This means **my return will be less than I put in**. Read that again. **Your principal can be penalized should you unstake before ½ your committed days are served OR if your stake length is fewer than 179 days due to the 90 days minimum penalty**.

Late Unstake

The system penalizes a user for leaving their Stake unattended after it has sat for its committed period. There is a grace period of 14 days and then upon ending the stake, the returned HEX will be reduced by 0.143% per day late (1% per late week). For example, if I have staked for 52 weeks, I have days 365 - 379 to end my Stake and have my HEX, earned interest, and bonus HEX returned to me. After that, my total payout will be penalized 0.143% per day. This includes Principal so if I'm 700 days late (just under 2 years), I will receive 0 HEX regardless of stake length or Principal.

Stake Example

This will be a simplified example to demonstrate how the contract flow works. It ignores the LargerPaysBetter bonus because I want to show small enough units to make sense, which means small, pointlessly confusing percentage bonuses. More detailed information will be available at https://bit.ly/hex-staking-gainz

- (A), (B), and (C) each claim HEX on the same day and by coincidence each end up with 10,000 HEX
- A stakes 10,000 HEX for 182 days (6 months), B stakes 10,000 HEX for 364 days (about 1 year), and C stakes 10,000 HEX for 1820 days (about 5 years)
- To simplify, we'll assume they are the only 3 stakers and they get a share rate of 1:1
- The contract converts HEX to shares using the LongerPaysBetter scalar (days/1820) resulting in
    - (A) 10,000 * (1 + 182/1820) = **11,000 shares**
    - (B) 10,000 * (1 + 364/1820) = **12,000 shares**
    - (C) 10,000 * (1 + 1820/1820) = **20,000 shares**
    - **Total Shares are 43,000**
- Every day a payout pool is accumulated from **inflation** * (1 + **Critical Mass** bonus + **Virality** bonus). Let's assume for simplicity that there are no further claims so Critical Mass, and Virality remain constant
- For this example assume inflation is **714.285 HEX/day**, **15% Critial Mass**, **25% Virality**, making daily payout 714.285 * (1 + 0.15 + 0.25) = 1,000 HEX/day
    - Day 1, say the pool is 1,000 HEX among 43,000 shares
    - Day 2, the pool is 1,000 HEX among 43,000 shares
    - Etc. until Day 182
- On day 182, (A) ends their stake without penalty
    - The contract goes through each day, 1 - 182
    - It accumulates a payout for (A) using (A)'s shares / total shares

        - **1000 HEX * 11,000 (A) shares / 43,000 total shares = 255.81 HEX**
    - It does this for all 182 days, resulting in **46,557.42 HEX** as the payout
    - This is minted to their address along with their **initial 10,000 HEX**
    - Their address now has **56,557.42 HEX**

This process will happen for (B) and (C) as well once their stakes mature, albeit for 364 and 1820 days' data.

Bonuses/Modifiers

All bonuses and modifiers I describe here are advertised on the website and coded within the contract. I saw all of them and validated the math and operator precedence using Solidity documentation (if that doesn't make sense to you, it means that I made sure it does what it claims to do).

Claim-related

These are listed in the order they are applied.

GoxMeNot: The website says it doesn't allow some well known addresses of bad actors to claim (e.g. Mt. Gox). *This will be achieved during the construction of the snapshot Merkle tree. Richard says he will publish sufficient information to allow users to build the snapshot Merkle tree for themselves so we can validate that the snapshot is accurate but no such information has yet been published and so I cannot verify that this happens at the moment.*

SillyWhalePenalty: When claiming, if the supplied BTC address held 1000 or more bitcoins at the time of snapshot, the claim is scaled down. The scaling goes from 50% for 1000 up to 75% for 10,000+ bitcoins. This scaling is linear so 5500 bitcoin claims (½ way between 1000 and 10,000) is scaled down 62.5%. This means a claim of 1000 BTC is only credited with 500 times 10k HEX. 10,000 BTC claims would only receive 25,000,000 HEX (2500 * 10,000 HEX/BTC). 5500 BTC claim would only receive 20,625,000 HEX. Pro Tip: This penalty can be avoided by splitting large addresses into multiple addresses each containing < 1000 BTC *before the snapshot occurs*.

LatePenalty: The system rewards fast claims and punishes slow claimers. The intent is to reduce the value of your claim by 0.286% per day (~2% per week), dropping you to 0 after day 351 of the project launch. The function is easy, it just multiplies your claim by (350 - days passed)/350. So on the first claim day, 0 days passed, you get **claim * 350/350**. Three weeks later you only receive **claim * 329/350**. For example 1 BTC claim would net 10,000 HEX on the first day and only 9400 HEX three weeks later. The earlier example of claiming 2 weeks after first claim day (May 19th), we would incur a 2,000 HEX penalty on our 50,000 claim as shown in the Claim Examples section.

Speed: In the Basic Flow section, I claimed 50,000 total HEX across 2 BTC addresses. There is a “Speed Bonus” applied to these claims. The bonus is applied for the first 350 claim days after launch and starts at 20% on the first day and drops by 0.057% each day thereafter. For example a 5 BTC claim on May 5th nets me the full 20% (5 * 10,000 HEX base + 10,000 HEX bonus), bringing my total to 60,000 HEX. In our example we claimed 2 weeks late so our base claim is now 48,000 HEX and the bonus has dropped to 19.2% of that new base value, or 9216 bonus HEX. If instead I claimed 175 days (6 months-ish) later, I both lose out on bonus and some of my claim has been redistributed via We Are All Satoshi. My base claim value would now be 25,000 HEX (**LatePenalty**) and the **Speed** bonus is only 10% or 2500 additional HEX.

Referrals: The website allows you to generate a referral link. This sets a cookie in the browser of the person who clicks it. The cookie merely states your ETH address and is read by the claim tool. The contract receives a “referrer” address and at the time of claim, pays the “referrer” address 20% of the claim value *including the Speed bonus*. Better still, it *pays the person using the link a 10% bonus for having a referrer*. To restate, using a referral link gives you 10% extra. Your referrer makes 20% on your total claim value. For example, if you claim 1 BTC on the first claim day, it will get a full Speed Bonus of 20%, resulting in 12,000 HEX. Clicking a referral link generates an additional 10% for you, 1200 HEX, and 20% for your referrer, 2640 HEX.

Self referring works, meaning you can generate a referral link for your own ETH address, click it, claim your HEX, and receive the 20% referral bonus. I mention this because it should be obvious that you could do this using a “dummy” ETH address to receive claim bonuses and your “real” address to claim against. Then consolidate or not at your leisure.

Stake Related

Modifies the Payout Pool

We Are All Satoshi

As part of the snapshot process, the total amount of bitcoin in existence is calculated. Each day after the first claim day for 350 days, 0.2857% of this total, minus claimed coins, is snapshotted. For example, let’s say there are 17,500,000 bitcoins total in the snapshot. At the end of the first day, ½ of the possible claims have been made, meaning 8,750,000 bitcoins are unclaimed. The next day, 25,000 bitcoins are taken from the 8,750,000 pool and recorded as “unclaimed”. The next day, assuming no more claims, another 25,000 bitcoins are drained and added to the unclaimed amount. And so on.

If more people claim, the daily unclaimed pool is reduced by their claim sizes. To be clear, their claim size is reduced at the same rate as coins are marked “unclaimed” so everything scales appropriately. Meaning, if I have 50 BTC and don’t claim for 35 days (5 weeks), I only get 45

BTC worth of HEX. The unclaimed pool has marked 5 of my BTC as unclaimed. *The math in the contract works such that claims and unclaimed payouts balance.*

After the claim phase ends, 352 days after contract launch, all tabulated unclaimed coins are paid to stakers. How this achieved in practice is if you have a stake that includes this special day, in addition to your interest payout you will receive a one time payout of your share of all unclaimed coins. For example, above we imagined that 8,750,000 BTC are claimed and no more. After the end of claim phase any active stake on that day will receive a one time slice of the unclaimed coins upon *stakeEnd*. If you have 5% of the shares of all stakers, you will receive an additional payout of **0.05 * 8,750,000 BTC * 10,000 HEX/BTC = 4,375,000,000 HEX**

Critical Mass/Virality

I'm listing these together because they work essentially the same and are calculated together in the contract. Critical Mass is a bonus applied to the payout pool equal to coins claimed / total possible coins. So if there are 17,500,000 total claimable bitcoins and 13,125,000 have been claimed, the bonus is 75%. Virality is similar but for bitcoin addresses. It is the number of eligible addresses claimed / total eligible addresses. The bonuses are additive, meaning calculated separately and both added to the payout (payout * (1 + coins claimed/total coins + addresses claimed/total addresses)).

Early/Late Unstake

These aren't bonuses but rather penalties incurred by other stakers. As described in the Unstaking Gotchas section, there are actions a user can take which incur penalties. Half of all HEX removed from the staker's return by penalties are added to the payout pool on the day the stake is ended or "good accounting" has been run. For example, say a stake ends early and it incurs a 10,000 HEX penalty. They do this on day 910. The interest pool for active stakers on day 910 is increased by 5,000 HEX. Then on day 1050, a stake that is late has "good accounting" run on it and 15,000 HEX are lost to a late penalty. That day's payout pool would have 7500 HEX added. And so on.

Modifies Your Stake

These bonuses are how your HEX are multiplied to become Shares. The bonus for LongerPaysBetter caps at 2x, meaning 10ish years or 3641 days, and the bonus for BiggerPaysBetter caps at 10% for 150,000,000 HEX stakes. This means that with a 10 year stake, your Shares are equal to triple your coins. With a 75,000,000 HEX stake, you receive a 5% bonus on top of that. The total after bonuses is the number used to compute Shares.

LongerPaysBetter

The longer you stake, the more you make. The formula is **(days staked - 1) / 1820**. It looks complicated but comes out 20% per stake year (the contract uses 364 days/year). The "minus 1" accounts for the minimum stake period of 1 day.

BiggerPaysBetter

The larger your stake, the more you make. The bonus formula is **HEX/150*10^7**, capping at 10% for stakes of 150,000,000 HEX. **Note: the percentage is capped, the bonus HEX is not**.
Example Stakes

- 1,000,000 HEX = 0.0667% bonus, or 667 HEX bonus
- 10,000,000 HEX = 0.667% bonus, or 66,667 HEX bonus
- 100,000,000 HEX = 6.67% bonus, or 6,666,667 HEX bonus
- 200,000,000 HEX = 10.0% bonus, or 20,000,000 HEX bonus

The Origin Address

The contract specifies an ETH address as the Origin Address. This address is paid HEX by the contract in a few ways

- The Origin is paid ½ of all HEX reclaimed by penalties (the other half going to the payout pool)
- The Origin is paid a copy of all bonus payments
    - Speed claim bonus
    - Referral bonus
    - We Are All Satoshi increments
    - Critical Mass/Virality bonuses

Contract Functions

This section will be a little more technical and discuss the different functions exposed by the HEX contract and can be invoked by any caller willing to pay the gas. Some of these functions are marked `external` and some `public`. From the Solidity documentation there is no difference to a caller other than `external` functions are sometimes more gas efficient. I suspect this is used more as a marker by the developer of which functions are intended to be called on a regular basis by an external client.

External Functions

These are the `external` functions that are the primary normal interactions with the contract

Informational

Functions that related to information about the contract

globalInfo

This returns the global state of the contract in the form of an array of values

- lockedHeartsTotal
- nextStakeSharesTotal
- shareRate
- stakePenaltyPool
- daysStored
- stakeSharesTotal
- latestStakeId
- unclaimedSatoshisTotal
- claimedSatoshisTotal
- claimedBtcAddrCount
- currentContractDay
- totalSupply (i.e. circulating supply)

allocatedSupply

Returns the total supply of HEX in circulation plus the aggregate locked HEX. This is importantly different from the ERC-20 function *totalSupply* because the contract burns locked HEX, meaning it will not be accounted for by *totalSupply*.

totalSupply

Returns the total minted HEX which is synonymous with the circulating supply of HEX.

dailyDataUpdate

Takes as input a day#, with 0 being interpreted as “current day”. This idempotently calculates and sets the day’s payout data before the given day#. For example, 1 year after launch, I call this function with `343` as input and it will calculate all day stake data for days 1 - 342 or whichever days had not yet been calculated.

dailyDataRange

Takes as input a day# and number of days. This is a data fetching function that returns the list of payout data for the requested range of days. The data is packed into uint256 values (unclaimed satoshi snapshot << 160, share total << 80, payout total).

currentDay

Returns current contract day.

Transform

Functions to participate in Transform Lobbies and related functionality

xfLobbyEnter

Enters supplied ETH into the current day’s Transform Lobby. If a referrer address is provided, it is captured in order to credit that referrer with normal referral bonuses upon leaving the lobby

(collecting HEX). Multiple entries may be made in a day and are entered into a queue, resolved first-in-first-out upon “leave”.

xfLobbyExit

Takes a target day and number of entries to resolve. For example, you join day 3 lobby four times. After day 3, you may call this function with 3 for the day and 0-4 for the number of entries to resolve. 0 means “all”. The contract will resolve the supplied number of entries from oldest to newest, accumulating HEX to be minted to the caller and minting referral bonus HEX to referrers as entries are resolved.
In the above example say the first and second entry have different referrers, A and B. If I call *leaveXfLobby(3, 2)*, the contract will calculate my transformed HEX for the first entry, mint the referral bonus to A, calculate my transformed HEX for the second entry, mint the referral bonus to B, then mint the total transformed HEX to me.

xfLobbyflush

Flushes transformed ETH to predefined flush address.

xfLobbyRange

Returns the lobby values for a range of days specified by begin and end day inputs

xfLobbyEntry

Returns the ETH and referrer for a given entry on a given day. The input “entryId” is a bit-packed value of the day + index in the entry queue.

xfLobbyPendingDays

Returns an array of days which have pending entries eligible for collection.

Stake

Functions to stake or directly related to staking

stakeStart

Begins a stake. Takes as inputs the stake amount and stake length. The contract calculates shares based on stake length and adds a stake entry for the user. The identifier for this stake is a just a number equal to the number of stakes in the system plus 1. For example, I have 3 active stakes and other users have a combined 20 active stakes. The global “next stake id” is 24 and start a new stake. To refer to the new stake later, its id is 24. Starting a stake burns the committed HEX and accounts for them in the global “allocated supply”.

stakeEnd

Ends a stake. The logic changes quite a bit if the stake ends early, late, or on time. The details are covered above. The necessary inputs are an id for the stake (this was determined when the

stake was created) and the index of the stake among the user's active stakes. This mints new HEX to fulfill the stake return and adds to the total (i.e. circulating) supply.

stakeGoodAccounting

This function safely ends a mature stake. If the stake is not mature, the function errors. It does *not* pay out to the staker. It can be invoked by anyone on behalf of any staker. It takes as input a staker address and a stake id for that address.

stakeCount

This returns the number of open stakes a user has.

Claim

Functions to claim or directly related to claiming

btcAddressIsClaimable

Takes a BTC address and returns whether it can be claimed, meaning it has not been claimed yet and is valid

btcAddressIsValid

Validates whether claim parameters BTC address, satoshis, and proof constitute a valid claim

merkleProofIsValid

Validates a Merkle proof and Leaf are valid for the Merkle Tree built from the UTXO set

btcAddressWasClaimed

Takes a BTC address and returns whether it has been claimed or not.

claimBtcAddress

This is a big one. This is the claim function that could be called by a naive client but certainly is intended to be invoked by the claim tool. It takes several inputs to validate a bitcoin claim and returns the resulting "free" HEX including the speed bonus, late penalty, and referral bonus (10% if other-referred, 32% if a self-referral). "Free" has a special meaning here, see below.

For folks interested in the technical details, the inputs are:

- `referrer` is the referring ETH address, if any
- `v`, `r`, and `s` are known parameters needed for the ECDSA signature validation
- `addrType`, `pubKeyX`, and `pubKeyY` are parameters to convert a public ECDSA key into its associated BTC address

- `claimToAddr` is the destination ETH address that shall receive the HEX should the claim be valid
- `proof` is the data used to prove that a given address is in the snapshot (`verifyProof` function). The details here are technical, but rest assured this is standard and in fact one of the benefits of the data structures used for BTC data
- `rawSatoshis` the claim amount in Satoshis, the smallest denomination of BTC (think pennies to bitcoin's dollar)
- `autoStakeDays` - this is unrelated to the claim verification but rather a new contract Rule, explained below

The contract now automatically stakes 90% of a claim's value for a minimum of 350 days. The function takes autostake days as an input and validates that it is as least 350 days. Emergency unstaking is disabled before 350 days. For example, a "default" claim with 350 day autostake will be allowed to end that stake at maturity but not before. An aggressive user may claim with supplied autostake days of 700. After 350 served days, they may unstake albeit with all applicable early unstake penalties.
The remaining 10% of claimed HEX is minted to the claimant and may be traded or staked as desired.

Public Functions

These are the `public` functions that the contract exposes to be called but may or may not make sense. This statement will make more sense shortly.

claimMessageMatchesSignature

Synthesizes a claim message from a key then compares the extracted address from the recreated message and the address converted from the public key

pubKeyToEthAddress

Converts a public key to an ETH address

pubKeyToBtcAddress

Converts a public key to a BTC address

Contract Events

The Events have been restructured in the contract to use custom bit-packing so these values are not correct. The values they capture are all documented below with roughly the sizes indicated.

Bit packing means that the author of the contract and wallet uses only uint256 values and has an encoding scheme that fills in the 256 bits in predictable chunks. This is because Solidity only allows uint sizes in increments of 8 bits, but many values in HEX require numbers close to a multiple of 8. This means that some gas efficiency can be gained by using more precise sizing and encoding the values into uint256 values.

XfLobbyEnter

Emitted upon joining a daily lobby

Fields

uint40 timestamp,
address indexed memberAddr,
uint256 indexed entryId,
uint96 rawAmount,
address indexed referrerAddr

XfLobbyExit

Emitted upon resolving a lobby entry. For xfLobbyExit calls of multiple entries, multiple events are emitted.

Fields

uint40 timestamp,
address indexed memberAddr,
uint256 indexed entryId,
uint72 xfAmount,
address indexed referrerAddr

DailyDataUpdate

Emitted upon completion of daily data updates. A single event is emitted per batch of daily data computed.

Fields

uint40 timestamp,
uint16 daysStoredAdded,
uint16 daysStoredTotal,
bool isAutoUpdate,
address indexed updaterAddr

Claim

Emitted upon completion of a BTC claim

Fields

uint40 timestamp,
address indexed claimToAddr,
bytes20 indexed btcAddr,
uint8 claimFlags,
uint56 rawSatoshis,
uint56 adjSatoshis,
uint72 claimedHearts,
address indexed referrerAddr,
address senderAddr

ClaimAssist

Emitted upon completion of a BTC claim wherein the claimant is not the sender of contract call. This is emitted in addition to the above Claim event.

Fields

uint40 timestamp,
address claimToAddr,
bytes20 btcAddr,
uint8 claimFlags,
uint56 rawSatoshis,
uint56 adjSatoshis,
uint72 claimedHearts,
address referrerAddr,
address indexed senderAddr

StakeStart

Emitted upon starting a stake.

Fields

uint40 timestamp,
address indexed stakerAddr,
uint40 indexed stakeId,
uint72 stakedHearts,
uint72 stakeShares,
uint16 stakedDays,
bool isAutoStake

StakeGoodAccounting

Emitted upon call to *stakeGoodAccounting*, if successful.

Fields

uint40 timestamp,
address indexed stakerAddr,
uint40 indexed stakeId,
uint72 payout,
uint72 penalty,
address indexed senderAddr

StakeEnd

Emitted upon ending a stake

Fields

uint40 timestamp,
address indexed stakerAddr,
uint40 indexed stakeId,
uint72 payout,
uint72 penalty,
uint16 servedDays

ShareRateChange

Emitted if the share rate is changed by an *stakeEnd* with the new share rate.

Fields

uint40 timestamp,
uint40 shareRate,
uint40 indexed stakeId