xBTC Stablecoin System - Technical Documentation

Update - Feb 15, 2022:

We have developed and tested the contracts implementing EVI DAO governance functionalities, and xBTC stability fee functionality. xBTC debts now accrue interest, at a rate that can be set by governance (EVID). The governance also controls the treasury, where the surplus generated by stability fees is deposited. The contracts have been deployed on RSK testnet.[1] We have also initiated the process of auditing the EVI DAO contracts by requesting a quote from a few leading audit companies. The way these contracts work is described in more detail below, including links to the github codebase with working tests.

Update - Jan 5, 2022:

The core contracts allowing the functionality stated above have been deployed on RSK testnet.

- CDP creation can be done by depositing RBTC collateral using the `depositCollateral` function.
- Stablecoin minting mechanism - after depositing collateral, users can mint a certain amount of xBTC stablecoin,[2] using on the autonomous `computeDebtLimit` and `takedebt` functions available in the CDP contract.
- Stablecoin redemption mechanism - any holder of xBTC stablecoin can 'redeem' it for the underlying RBTC collateral at the prevailing redemption rate (which is calculated using SMA4), using the `redeem` function,
- CDP closure by repaying Debt to retrieve collateral - Only the original creator of the specific CDP can call these functions. After the full debt (xBTC) has been returned using the `returnDebt` function, the user can retrieve some or the entire RBTC collateral using the `removeCollateral` function.

CDPs in xBTC are analogous to SAFE functionality in RAI. Note the difference between *redemption*, which can be done by any holder of xBTC at any time, and *removeCollateral*, which can only be called by the CDP creator, *after* all xBTC debt has been repaid. Redemption happens at the prevailing redemption rate, whereas "remove collateral" simply returns the entire collateral that was originally deposited into the CDP. The way these

[1] See *C. Contract Address*, p5.
[2] EVI stablecoin is currently renamed xBTC stablecoin to represent its close relationship with BTC (1:1 collateralisation).

functions work is described in more detail below, *including links to the github codebase with working tests*.

A. Components

You can find the actual contract code, with tests of all the functionalities on our Github Repo-
For the xBTC CDPs and Stability fee model, see:
https://github.com/FoundationCryptoLabs/XSS/tree/governance
For the Governance contracts, see:
https://github.com/FoundationCryptoLabs/EVI-governance/tree/docs

I. Core CDP functionalities

CDPtracker.sol

- `function setCoin(address STABLE) external isAuthorized {}`
  ADMIN: Set xBTC coin address for calling mint/burn functions.
- `function depositCollateral() public payable {}`
  The CDP user sends a transaction to deposit collateral for minting xBTC. At this point the CDP is considered collateralized.
- `function takedebt(uint256 amount) public {}`
  The CDP user then sends a transaction to mint the amount of xBTC they want from CDP and in return the CDP accrues an equal amount of debt, locking them out of access to the collateral until the outstanding debt is paid. This transaction succeeds only if their debt limit is > or = the amount of xBTC requested. Debt limit depends on the amount of collateral deposited; it is calculated autonomously using the *`computeDebtLimit`* internal function. The initial debt limit is set such that 1.25BTC collateral must be deposited to mint 1 xBTC (125% collateral ratio).
- `function returndebt(uint256 amount) public {}`
  When the user wants to retrieve their collateral, they have to pay down their debt in the CDP, (plus the stability fee that continuously accrue on the debt over time).[3] Once the user sends the requisite xBTC to the CDP, paying down the debt and stability fee, the CDP becomes debt free. The amount of xBTC to return is currently the same as the original principal deposited; i.e. the interest rate is set to 0%. This parameter will be editable by governance.
- `function redeemCoins(uint256 amount) public {}`
  Any holder of xBTC can redeem it for the underlying collateral at the current redemption rate which is based on the SMA1458 of the BTC/USD exchange rate.

[3] Stability fee functionality not enabled yet.

This function calls the *peekSMA* and *peekBX* function from the Oracle contract, to get the current value of the redemption rate. This value is converted to a ratio of xBTC:BTC. For example, if SMA is 20000 and BX is 60000, the ratio is 3:1. Thus, 3 xBTC can be redeemed for 1 RBTC at that time.

- `function removeCollateral(uint256 amount) public payable{}`
  With the debt and stability fee paid down, the CDP user can freely retrieve all or some of their collateral back to their wallet by sending a transaction to the CDPtracker contract calling the removeCollateral function.

II. Stablecoin

Standard Authorized ERC20 functionality from GEB protocol; unchanged.

III. Oracle

- `function setCollateralRatio(uint256 ratio) external {}`

ONLY ADMIN/GOVERNANCE can call this: Change base collateral ratio for CDPs.

- `function peekCollateralRatio() external returns(uint256) {}`

Set by default to 12500 (125%).

- `function peekSMA() external returns(uint256){}`

Calculates SMA1458 of the BTC/USD exchange rate. Set to 20000 by default.

- `Function peekBX() external returns(uint256){}`

Returns the current BTC/USD exchange rate. Set to 60000 by default.

IV. xBTC stability fee

A fundamental feature of the EVID system is to accumulate stability fees on SAFE debt balances. This surplus provides revenue to the EVI Treasury, and gives EVID a secondary tool to adjust demand and supply and stabilize prices. The mechanism used to perform these accumulation functions is subject to an important constraint: ***accumulation must be a constant-time operation with respect to the number of Vaults***. Otherwise, accumulation events would be very gas-inefficient (and might even exceed block gas limits).

The solution is as follows: store and update a global "*accumulated rate*" (AR) value, which can then be multiplied by a normalized debt or deposit amount to give the total debt or deposit amount when needed.

Background:

Mathematically, this means we compute one AR variable that implicitly contains information about past interest rates, as:

$$AR = (1 + i)^t$$

i = stability fee (per second interest rate)

t = time elapsed (in seconds)

AR is thus a single, steadily increasing integer stored in the CDP contract. This is referenced by the SAFE vault of every user, *every time* any action is taken to update the SAFE. The SAFE-specific interest due is then calculated using the ratio:

$$Amount = nAR/lAR * Principal$$

nAR = new AR

lAR = AR at the time of the last update.

xBTC utilizes a modified version of the accumulated rates mechanism used in DAI/RAI to calculate current surplus, *without needing to keep track of historical balances and interest rates*.[4]

Code:

Calling TaxCollector.updateAR() computes an update to the AR based on the per second interest rate, and the time since updateAR was last called.
Then the TaxCollector invokes CDP.newAR(int new_rate) and CDP.issueSurplus(address treasury, int new_rate) which:

- adds new_rate to the RATE variable in CDP.
- increases the System's surplus by globalDebt*rate_ratio
- increases the system's total debt (i.e. issued xBTC) by globalDebt*rate_change.
- Mints the surplus xBTC tokens to the treasury.

The next step is to calculate a user's specific interest dues. To do this, the CDPTracker contract has the function updateUserDebt. rate_Ratio is calculated as the rdiv(newAR,originalAR) which gives us net interest due when multiplied by issuedDebt or globalDebt.

[4] For a complete explanation of the interest rate calculation model, see - https://docs.makerdao.com/smart-contract-modules/rates-module

Each individual Vault (represented by an SAFE struct in CDPTracker) stores a "normalized debt" parameter called debtIssued, and an originRate parameter with the AR at the last update. Any time it is needed by the code, the Vault's total debt, including stability fees, can be calculated as rmul(debtIssued,rate_ratio) . Thus an update to the AR (CDP.RATE) via TaxCollector.updateAR() effectively updates the debt for all Vaults collateralized with RBTC.

Note on Tests for Interest rate mechanism:
To facilitate testing of the interest rate model, we have hardcoded a time period of 86,400 seconds (1 day) into the updateAR0 & updateAR1, used to calculate interest due. This will be replaced by a `block.timestamp - InitTime` calculation, as mentioned in the comments.

Stability fee is set to : 1000000564701133626865910626 [which is a RAY format number, i.e. 10^-27 digit representation]. This computes to an interest rate of 5% per day. Once again, this was chosen for clarity in testing, the final version will utilize a governance-determined interest rate.

V. Governance

We utilize customized versions of the highly battle-tested governance contracts developed by Openzepplin - Specifically the GovernorVotes contracts from OZ v0.4 - which is an extension of Governor for voting weight extraction from an ERC20Votes token.[5] These contracts are compatible with the TALLY[6] governance frontend, which we will be setting up in the next milestone for ease in conducting governance activities including proposal, voting, and monitoring. Some key parameters of the Governor are highlighted below.

Governor.sol

Voting power is defined using the GovernorVotes module, which hooks to an ERC20Votes instance to determine the voting power of an account based on the token balance they hold when a proposal becomes active.
Quorum is set to 4% and is set by GovernorVotesQuorumFraction which works together with ERC20Votes to define quorum as a percentage of the total supply at the block a proposal's voting power is retrieved.
Voters are offered 3 options by GovernorCountingSimple: For, Against, and Abstain, and where only For and Abstain votes are counted towards quorum.
Proposal Threshold is set to 10,000 EVI tokens. Block time has been set to 30s based on RSK averages.

---

[5] For full documentation see - https://docs.openzeppelin.com/contracts/4.x/api/governance#GovernorVotes
[6] https://www.withtally.com

EVI Token

The voting power of each account in our governance setup will be determined by the EVI ERC20 token. The token implements the ERC20Votes extension. This extension will keep track of historical balances so that voting power is retrieved from past snapshots rather than current balance, which is an important protection that prevents double voting.[7] Governance Token distribution strategies will soon be finalised.

VI. Liquidation, Surplus & Deficit Auctions

There are three types of auctions used in MCD/GEB protocols:

1. Liquidation Auctions, where collateral is automatically liquidated in case the required collateral ratio is breached.
2. Surplus auctions that benefit governance token holders, by selling surplus to buy EVI tokens and burn them.
3. Deficit auctions where governance tokens are sold to make up for shortage of collateral, are two key auctions we're considering adding.

We have kept (2) & (3) on our roadmap, for their incentive aligning features between the EVI DAO and the zBTC stablecoin, and as its position as collateral of last resort. Due to unchanging collateral ratio requirements, unlike USD denominated collateral protocols like MCD or GEB, there appears to be no need for automatic liquidation functions (1) in cases of volatility for zBTC. However we may update that in case further analysis proves its utility.

B. Contract Addresses on Testnet:

The contracts enabling the above functionality are live on RSK testnet, and can be viewed on these addresses:[8]

CDPTracker [new, w/ interest rate model] :

https://explorer.testnet.rsk.co/address/0x7a0984E49418759Ef975F7d1d93f5606A8055b38

***EVIGovernor*:**

https://explorer.testnet.rsk.co/address/0x7E6d5340e848F218F157d6A6bc0B156f55454099

EVIToken*:*

https://explorer.testnet.rsk.co/address/0xD98f6Fb780A60f8993997a1F492A52A09F8b50AC

---

[7] ERC20votes token from OZ v0.4

[8] Running tests using public testnet nodes often results in "poolingblockerror" or "too many requests" error due to rate limits. Ideally, tests can be run using a private RSKJ Testnet node running on localhost, to support high latency. If unavailable, the contracts can equivalently be tested locally, using ganache.

***Coin*:**
https://explorer.testnet.rsk.co/address/0x3bf9e5bb65c580fbe1936bd7edd60aaad4f38eb0
***Oracle*:**
https://explorer.testnet.rsk.co/address/0x2Ef2757bD2c469a7F8afa251f17700aaa6B9F3B7

C. General Information about Codebase

xBTC uses a significantly adapted codebase derived from GEB, which itself is a fork of MCD. XSS has the following key differences to GEB:

- An autonomous algorithmic mechanism to resist inflation - in contrast with RAI which merely allows redemption rate to change upon "consensus of users", which has been hard to achieve in practice.
- Potential for significant value appreciation, as redemption rate is pegged to the Simple Moving Average of BTC/USD price.
- Collateral requirements denominated in BTC instead of USD - this eliminates the need for automatic liquidations, while maintaining collateral sufficiency using a probabilistic model.
- Higher incentives to deposit collateral and mint xBTC, as there is *no risk of collateral being liquidated.*[9]
- XSR savings rate set by Governance can be set higher as the risk of liquidation is now eliminated - increasing long term protocol revenues and reserves.

D. Next Steps

As of March, 2022, all the core contracts for deploying xBTC Stablecoin system are complete. Audits of the code are in progress. Further work revolves around building a web3 compatible frontend to:

1. Facilitate users in Opening & Closing CDPs, and redeeming xBTC tokens for the underlying collateral.
2. Conduct Governance Activities - using frontend like TALLY.

Moreover, we plan on focusing increasingly on community building and awareness as we approach launch, using multiple channels including social media and building a website + a blog. Upon completion of these steps, we will be ready for a full protocol launch. The estimated time for launch is early April.

---

[9] This is subject to the backtested assumption that BTC/USD price never falls below 66% of the SMA4 at any given point of time. Over the past 10 years, BTC/USD price has never fallen below 90% of SMA4.