USDx Design

Edit History

Time	Content	Author	Version	Note
Jun 3, 2019	Draft	Snow & Horsen	V1.1	
Jun 6, 2019	Add "All-in-one Mintage"	Snow	V1.2	
Jun 17, 2019	Modify the protocol framework	Horsen	V1.3	Add "constituent stablecoin Wrap" function
Aug 26, 2020	Add dToken description	Snow	V1.4	Describe constituent stablecoins converting to dToken in mintage
Aug 28, 2020	Edition	Snow	V1.5	
Aug 29, 2020	Modify "Mintage Process"	Snow	V1.6	
Sep 1, 2020	Revision	Margaret & Snow	V1.7	

USDx Introduction

USDx is a synthetic indexed stablecoin of ERC20 standard, which is pegged into a basket of USD stablecoins and is deployed on Ethereum Mainnet.

- USDx can be minted with a basket of selected USD stablecoins through decentralized smart contracts at a pre-determined weighting;
- Constituent stablecoins will be automatically converted to dToken on dForce Yield Markets to farm the most favorable interests from lending protocol;

- The initial reserve was comprised of four constituents with 30% USDC, 30% TUSD, 30% PAX and 10% DAI (1 USDx = 0.3 USDC + 0.3 PAX + 0.3 TUSD + and 0.1 DAI). Later, we removed DAI from the basket and adjusted its constituent weighting to include 80% USDC, 10% PAX, 10% TUSD (1 USDx = 80% USDC +10% PAX + 10% TUSD).
- Section refers to constituent stablecoins and their corresponding weightings, which can be adjusted;
  - In principle, fees associated with USDx mintage and disaggregation can be paid by either USDx or DF (dForce governance token), while disaggregation consumes DF only at present.
  -

Basic Concepts

- **Section**

  Section is a list of constituent stablecoins required to facilitate mintage and disaggregation of USDx, as well as the minimum consumption required for each constituent. Adjustments to constituent takes immediate effect, while disaggregation will follow the new weighting upon drainage of USDx minted out of the old weightings.

- **Mintage**

  USDx mintage and distribution will be triggered when reserves sitting in the pending pool satisfy the minimum consumption requirement for each constituent. Constituents deposited will be converted to dTokens and supplied to dForce Yield Market to earn yield, which will be allocated to USR for distribution to USR token holders (who deposit USDx into the USR contract).

- **All-in-one Mintage**

  All-in-one mintage enables users to deposit constituent stablecoins in accordance with the section to generate USDx. USDx will be allocated to depositors, while constituent remainders yet to satisfy the mintage condition will be locked in a smart contract (constituents pool) pending for mintage, or be withdrawn anytime by depositors.

- **Disaggregation**

The redemption of USDx and return of a basket of constituents will facilitate a disaggregation. Pro rata dToken will be converted to constituents.

- **Claim**

  Mintage only respond to distribution of USDx to depositor who triggered the mintage. Those who deposited constituent but failed to satisfy the mintage condition (with constituent sitting in the pending pool) can obtain USDx through Claim. USDx ready-to-claim can also be obtained by those who initiated new mintage request (on a 'first-come-first-serve' basis).

- **Withdraw**

  Constituent stablecoins pending for mintage can be withdrawn anytime by depositors.

- **Decimal**

  Minimal decimal of disaggregation serves to ensure the return of constituents out of USDx disaggregation, will not suffer a decimal loss produced by constituents' decimals. Put it simple, the amount of USDx burned at a time must be an integer multiple of the minimal decimal of disaggregation.

Protocol Structure

The illustration below explains the protocol structure for USDx 1.0 (including USR):

- **Permission Management**

  The owner manages call permissions for all contracts through DSGuard contract. Owner has the ultimate call permission for all restricted functions. DFPool, DFCollateral, DFFunds, and DFStore contracts can be called by DFEngine. The Mint/Burn function of USDx protocol can also be called DFEngine. DFEngine call be called by DFProtocol.

- **Contract Upgrade**

  DFProtocol/DFSetting/DFProtocolView designates USDx interfaces and is non-upgradable.

  DFCollateral/DFFunds is non-upgradable.

DFPool is implemented by Proxy and Implementation, with logic upgradable.

DFStore stores the state and data of the contract and does not support upgrades.

DFEngine is implemented by Proxy and Implementation, with logic upgradable. It can also be replaced by requestImplChange and confirmImplChange interface of DFProtocol.

- **ERC-20 Contract**

  USDx and DF follow ERC-20 standard, supporting Mint and Burn functions and is non-upgradable after deployment.

- **Wrapped ERC-20 Contract**

  Wrapped ERC-20 contracts add support to Wrap and UnWrap functions by generating a corresponding Wrapped ERC-20 token for each constituent stablecoin, providing a solution that tackles the decimal problem of constituents.

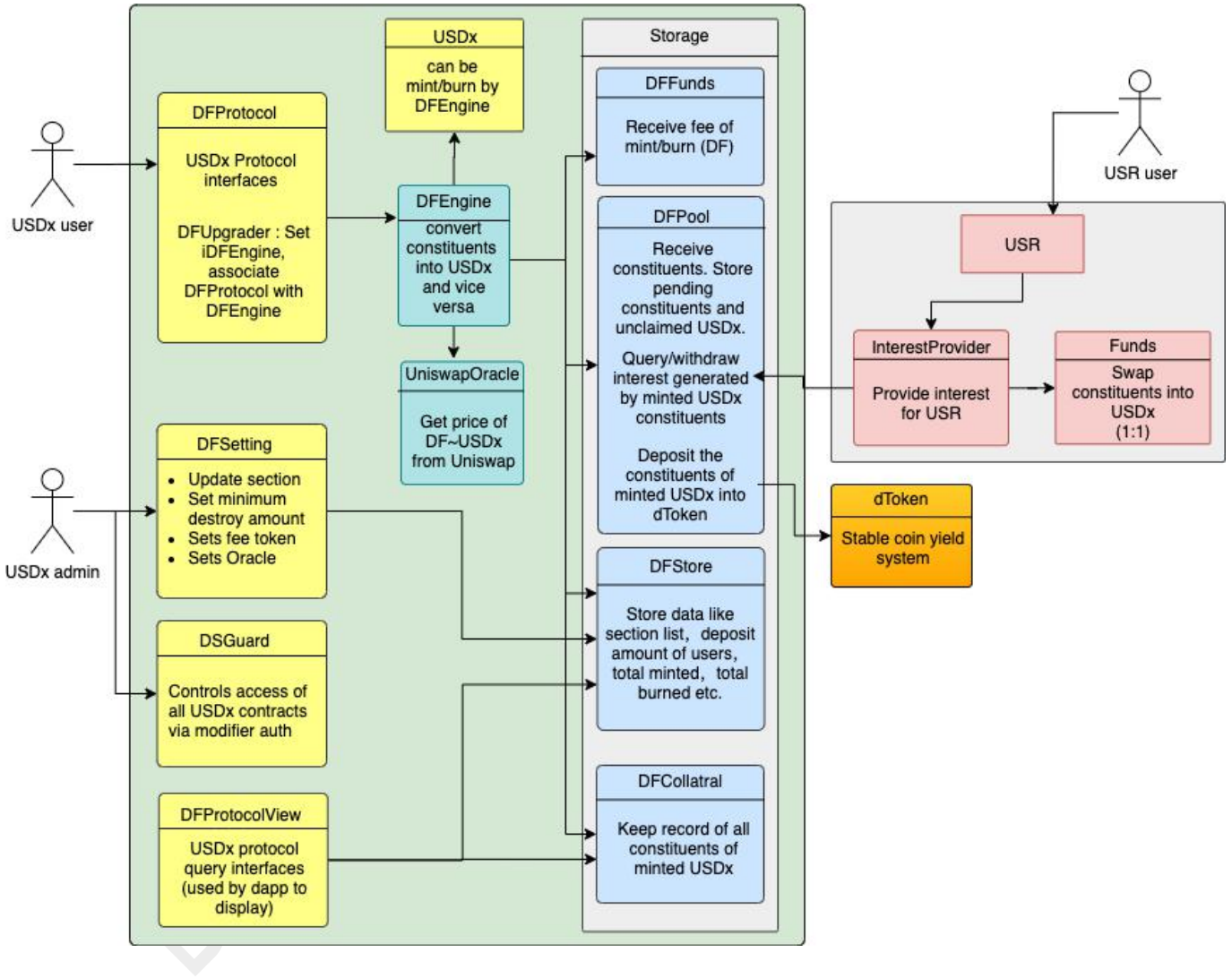
Function Design

1. **Mintage (from constituent stablecoins to USDx)**

   - When a user deposits one or multiple constituent stablecoins into the contract, certain amount of USDx or DF will be consumed (DF amount = USDx amount * mintage rate / DF price offered by Oracle) to facilitate the mintage of USDx, which can be claimed by any constituent depositor. One who triggered the mintage will receive USDx immediately but others need to claim to receive USDx. Simultaneously, constituent stablecoins pegged minted USDx will be converted to dToken, as well as constituent remainders yet to satisfy the mintage condition, will be stored in DFPool protocol.

   - Contributors of the same type of constituent are competing for minted and ready-to-claim USDx. Contributors of different types of constituents will not enter into the competition. When the mintage is triggered, USDx will be allocated to constituent stablecoin contributors based on weighting defined by Section, with one who triggers the mintage receiving USDx automatically. In the event there are remainders pending for mintage, users can choose to withdraw constituents or wait to receive USDx by competing for claim. Please note contributors of the same type of constituent need to compete for claim until all inventories are drained.

2. Disaggregation (from USDx to constituent stablecoins)


USDx user
USDx admin
USR user
DFProtocol
USDx Protocol interfaces
DFUpgrader : Set iDFEngine, associate DFProtocol with DFEngine
USDx
can be mint/burn by DFEngine
DFEngine
convert constituents into USDx and vice versa
UniswapOracle
Get price of DF~USDx from Uniswap
DFSetting
Update section
Set minimum destroy amount
Sets fee token
Sets Oracle
DSGuard
Controls access of all USDx contracts via modifier auth
DFProtocolView
USDx protocol query interfaces (used by dapp to display)
Storage
DFFunds
Receive fee of mint/burn (DF)
DFPool
Receive constituents. Store pending constituents and unclaimed USDx.
Query/withdraw interest generated by minted USDx constituents
Deposit the constituents of minted USDx into dToken
DFStore
Store data like section list, deposit amount of users, total minted, total burned etc.
DFCollatral
Keep record of all constituents of minted USDx
USR
InterestProvider
Provide interest for USR
Funds
Swap constituents into USDx (1:1)
dToken
Stable coin yield system

When a user redeems USDx, certain amount of USDx or DF will be consumed (DF amount = USDx amount * mintage rate / DF price offered by Oracle) to facilitate the redemption. USDx will be disaggregated into constituent stablecoins at a pre-determined weighting and returned to the user based on the disaggregation weighting defined by Section, which will trigger the burning of dToken into constituents.

3. **Mintage and disaggregation fees can be paid by either USDx or DF with independent rates.**

4. **Section management**

- The Section defines both USDx mintage and disaggregation. Certain amount of constituent stablecoins will be locked in smart contracts which is decided by the moment as well as amount of USDx to be minted or disaggregated. This should be taken into account to facilitate replacement of Section.

- Add Section: Multiple Sections can be added for USDx, which forms an orderly queue. The mintage will always follow the most up-to-date one, and record the total amount of minted USDx under each Section utilized. On the contrary, the disaggregation will stick to the foremost and switch to the next until all USDx minted out of previous Section are completely destoryed.

5. **Decimal of constituent stablecoins**

The decimal of constituent stablecoins may vary. To tackle this problem, we added DSWrappedToken.sol contract by wrapping constituent stablecoins into internal ERC20 tokens starting with 'x ' (hereinafter referred to as xToken). For example, when a user deposits USDC (decimal = 6), pro rata xUSDC of 18 decimals will be generated to facilitate calculation. Vice versa, when a user disaggregates USDx, pro rata amount of xUSDC will be calculated out of disaggregation weighting defined by Section, followed by destroy of xUSDC and return of pro rata USDC to the user.

Mintage Process

Assume the Section is {USDC:30, PAX:30, TUSD:30, DAI:10}, representing 30%-30%-30%-10% weighting for mintage. Simultaneously, the minimum constituent stablecoin consumption is also 30-30-30-10 (assuming that the four constituent stablecoins possess identical decimal). A, B, and C are three different users:

Tx No.	Action	Pending Pool	Constituents Locked	USDx allocation claimable change	User USDx Balance	Remarks
	A user DEPOSIT {10 USDC; 50 PAX; 50 TUSD; 50 DAI}	10 USDC	0 USDC	USDC: 0	Total: 0 {A: 0; B: 0; C: 0} {Claimable: 0}	Mintage criteria not satisfied. ---------- A user： Receive 0 USDx Constituents withdrawable: {10 USDC; 50 PAX; 50 TUSD; 50 DAI} USDx claimable: 0 USDx
		50 PAX	0 PAX	PAX: 0		
		50 TUSD	0 TUSD	TUSD: 0		
		50 DAI	0 DAI	DAI: 0		
2	B user DEPOSIT {50 USDC; 10 PAX}	60 -> 30 USDC	30 USDC	USDC: 30 -> 0	Total: 100 {A: 0; B: 40; C: 0} {Claimable: 60}	100 USDx minted. USDx allocation： {USDC: 30; PAX: 30; TUSD: 30; DAI: 10} ---------- B user: Receive 40 USDx (30 from USDC allocation and 10 from PAX allocation) Constituents withdrawable: {20 USDC; 0 PAX; 0 TUSD; 0 DAI} USDx claimable: 0 USDx ---------- A user： Receive 0 USDx Constituents withdrawable: {10 USDC; 30 PAX; 20 TUSD; 40 DAI} USDx claimable: 60 USDx (claimable: 0 USDC; 20 PAX; 30 TUSD; 10 DAI)
		60 -> 30 PAX	30 PAX	PAX: 30 -> 20		
		50 - > 20 TUSD	30 TUSD	TUSD: 30		
		50 - > 40 DAI	10 DAI	DAI: 10		
3	C user DEPOSIT {20 PAX}	30 USDC	30 USDC	USDC: 0	Total: 100 {A: 0; B: 40; C: 20} {Claimable: 40}	Mintage criteria not satisfied. C user: Receive 20 USDx（20 from PAX allocation） ---------- B user： Constituents withdrawable: {20 USDC; 0 PAX; 0 TUSD; 0 DAI} USDx claimable: 0 USDx ---------- A user： Receive 0 USDx Constituents withdrawable: {10 USDC; 50 PAX; 20 TUSD; 40 DAI} USDx claimable: 40 USDx (claimable: 0 USDC; 0 PAX; 30 TUSD; 10 DAI)
		30 - >50 PAX	30 PAX	PAX: 20 -> 0		
		20 TUSD	30 TUSD	TUSD: 30		
		40 DAI	10 DAI	DAI: 10		
4	B user WITHDRAW {15 USDC}	30 -> 15 USDC	30 USDC	USDC: 0	Total: 100 {A: 0; B: 40; C: 20} {Claimable: 40}	Mintage criteria not satisfied. ---------- B user： Constituents withdrawable: {5 USDC; 0 PAX; 0 TUSD; 0 DAI} USDx claimable: 0 USDx ---------- A user： Constituents withdrawable: {10 USDC; 50 PAX; 20 TUSD; 40 DAI} USDx claimable: 40 USDx (claimable: 0 USDC; 0 PAX; 30 TUSD; 10 DAI)
		50 PAX	30 PAX	PAX: 0		
		20 TUSD	30 TUSD	TUSD: 30		
		40 DAI	10 DAI	DAI: 10		
5	A user CLAIM	15 USDC	30 USDC	USDC: 0	Total: 100 {A: 40; B: 40; C: 20} {Claimable: 0}	Mintage criteria not satisfied. ---------- B user： Constituents withdrawable: : {5 USDC; 0 PAX; 0 TUSD; 0 DAI} USDx claimable: 0 USDx ---------- A user： Receive 40 USDx Constituents withdrawable: : {10 USDC; 50 PAX; 20 TUSD; 40 DAI} USDx claimable: 0 USDx (claimable: 0 USDC; 0 PAX; 0 TUSD; 0 DAI)
		50 PAX	30 PAX	PAX: 0		
		20 TUSD	30 TUSD	TUSD: 30 -> 0		
		40 DAI	10 DAI	DAI: 10 -> 0		