About USDx

USDx is a synthetic and indexed stablecoin issued on the Ethereum. It is pegged into a basket of selected stablecoins (1 USDx = 80% USDC + 10% PAX + 10% TUSD) at a pre-determined weighting which can be adjusted via on-chain governance. Anyone can visit https://usdx.dforce.network to interact with USDx contract via web， one can also interact with USDx on contract level, here we will explain how.

Interface

User can interact with interfaces of DFProtocol and DFProtocolView.

Interfaces of DFProtocol

1. ***deposit(address _tokenID, uint _feeTokenIdx, uint _amount) return (uint)***

Description: deposit one of the constituents to mint USDx

Parameters:

Input:

_tokenID: address of constituent.

_feeTokenIdx: fee token id, only 0 is supported now.

_amount: amount of constituent to deposit, should be integer in shape of token decimal precision.

Output: USDx amount minted.

2. ***withdraw(address _tokenID, uint _feeTokenIdx, uint _amount) return (uint)***

Description: withdraw one of the constituents not minted yet.

Parameters:

Input:

_tokenID: address of constituent.

_feeTokenIdx: fee token id, only 0 is supported now.

_amount: amount of constituent to withdraw, should be integer in shape of token decimal precision.

Output: amount withdrew, 0 means nothing withdrew.

3. ***destroy(uint _feeTokenIdx, uint _amount)***

Description: redeem USDx, get all constituents.

Parameters:

Input:

_feeTokenIdx: fee token id, only 0 is supported now.

_amount: amount of USDx to redeem, should be integer in shape of token decimal precision.

Output: None.

4. claim(uint _feeTokenIdx) return (uint)

Description: claim USDx

Parameters:

Input:

1）_feeTokenIdx: fee token id, only 0 is supported now.

Output: amount of USDx claimed.

5. oneClickMinting(uint _feeTokenIdx, uint _amount)

Description: deposit all constituents requried in minting section and mint USDx

Parameters:

Input:

✓ _feeTokenIdxfee: token id, only 0 is supported now.

✓ _amount: USDx to be minted.

Output: None.

interface of DFProtocolView

1. getUSDXForDeposit(address _tokenID, uint _amount) returns (uint)

Description: get USDx amount to be minted when deposit one of the constituents；

Parameters:

Input:

1）_tokenID: address of constituent to be deposited.

2）_amount: amount to be deposited.

Output: amount to be mined.

2. getUserMaxToClaim() returns (uint)

Description: get maximal amount of USDx to be claimed by sender.

Parameters:

Input: None.

Output: amount of USDx to be claimed.

3. getColMaxClaim() returns (address[] tokenID, uint[] balance)

Description: get maximal amount of USDx to be claimed of constituents

Parameters:

Input: None.

Output:

1）tokenID: token address.

2）balance: list of amount of USDx to be claimed of constituents.

4. getMintingSection() returns (address[] tokenID, uint[] weight)

Description: get current mining section

Parameters:

Input: None.

Output:

tokenID: list of constituent addresses.

weight: weight of each constituen in current minting section.

5. getBurningSection() returns (address[] tokenID, uint[] weight)

Description: get current burning section

Parameters:

Input: None.

Output:

tokenID: list of constituent addresses.

weight: weight of each constituent in current burning section.

6. getUserWithdrawBalance() returns (address[] tokenID, uint[] balance)

Description: get amount of constituent available to withdraw
Parameters:

Input: None.

Output:

tokenID: list of constituent addresses.

balance: amount of each constituent available to withdraw.

7. getPrice(uint _tokenIdx) return (uint value)

Description: get price of token.

Parameters:

Input: _tokenIdx: token id for USDx, only 0 is supported now.

Output: price of token, number with 18 decimal.

8. getFeeRate(uint _processIdx) return(uint value)

Description: get fee rate of action

Parameters:

Input: processIdx: type of action, 0: deposit，1:destroy，2:claim，3:withdraw.

Output: molecule part of fee rate, denominator is 10,000.

9. getDestroyThreshold() returns (uint)

Description: get minimal precision of amount of destroying USDx, it means the amout to be destroyed should always to be integral multiple of this value.

Parameters:

Input: None.

Output: minimal value of USDx

About Token Approve

1) Before calling deposit function, for example, deposit USDC into USDx protocol, you should approve DFPool contract to transfer your USDC.

2) oneClickMinting requies all of the constituent approving to DFPool contract before calling.

3) Destroy function needs USDx and DF approve to DFEngine contract before calling.

Deployed Contracts

Mainnet	**Address**
PAX	0x8e870d67f660d95d5be530380d0ec0bd388289e1
TUSD	0x0000000000085d4780B73119b644AE5ecd22b376
USDC	0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48
DF	0x431ad2ff6a9c365805ebad47ee021148d6f7dbe0
USDx	0xeb269732ab75a6fd61ea60b06fe994cd32a83549
DFProtocol	0x5843f1ccc5baa448528eb0e8bc567cda7ed1a1e8
DFProtocolView	0x097Dd22173f0e382daE42baAEb9bDBC9fdf3396F
DFEngine	0x3ea496977A356024bE096c1068a57Bd0B92c7d7c
DFPool	0x7FdcDAd3b4a67e00D9fD5F22f4FD89a5fa4f57bA