USR Scheme

❖ **History**

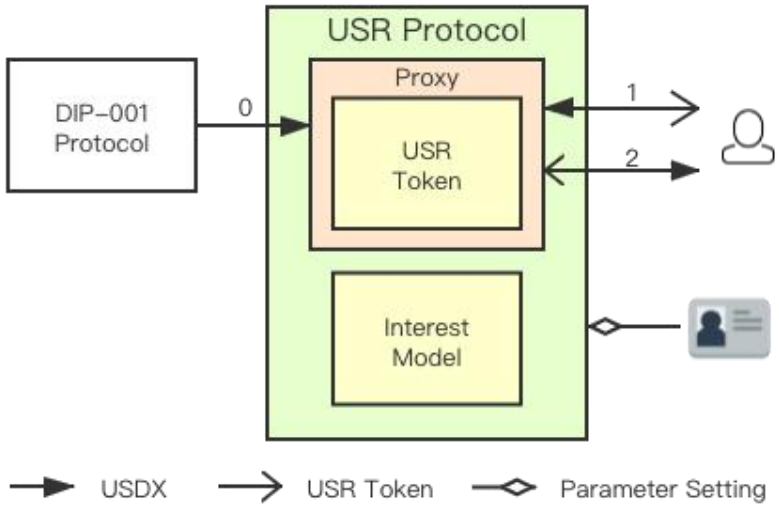
Date	Contents	Author
2020.2.25	v0.1	Horsen
2020.2.26	v0.2	Horsen
2020.2.27	v0.3	Skyge
2020.2.28	v0.4	Skyge
2020.3.4	v0.5	Skyge
2020.3.6	v0.6	Horsen
2020.3.9	v0.7	Skyge
2020.3.11	v0.8	Horsen

❖ What is USR

USDx Savings Rate (USR) is an addition of dForce Protocol that allows any USDx holder to earn risk-free savings. The savings paid out to USDx holders are financed by DIP-001 protocol which deposits constituent stable coins into the decentralized lending market to earn interest.

Users only need to deposit USDx into the USR (USDx Saving Rate) contract and then get "USR Token". The exchange rate between USR Token and USDx will continue to increase according to annualized interest, and users can use the USR token to exchange USDx from the USR contract at any time to obtain the principal and interest.

❖ Architecture



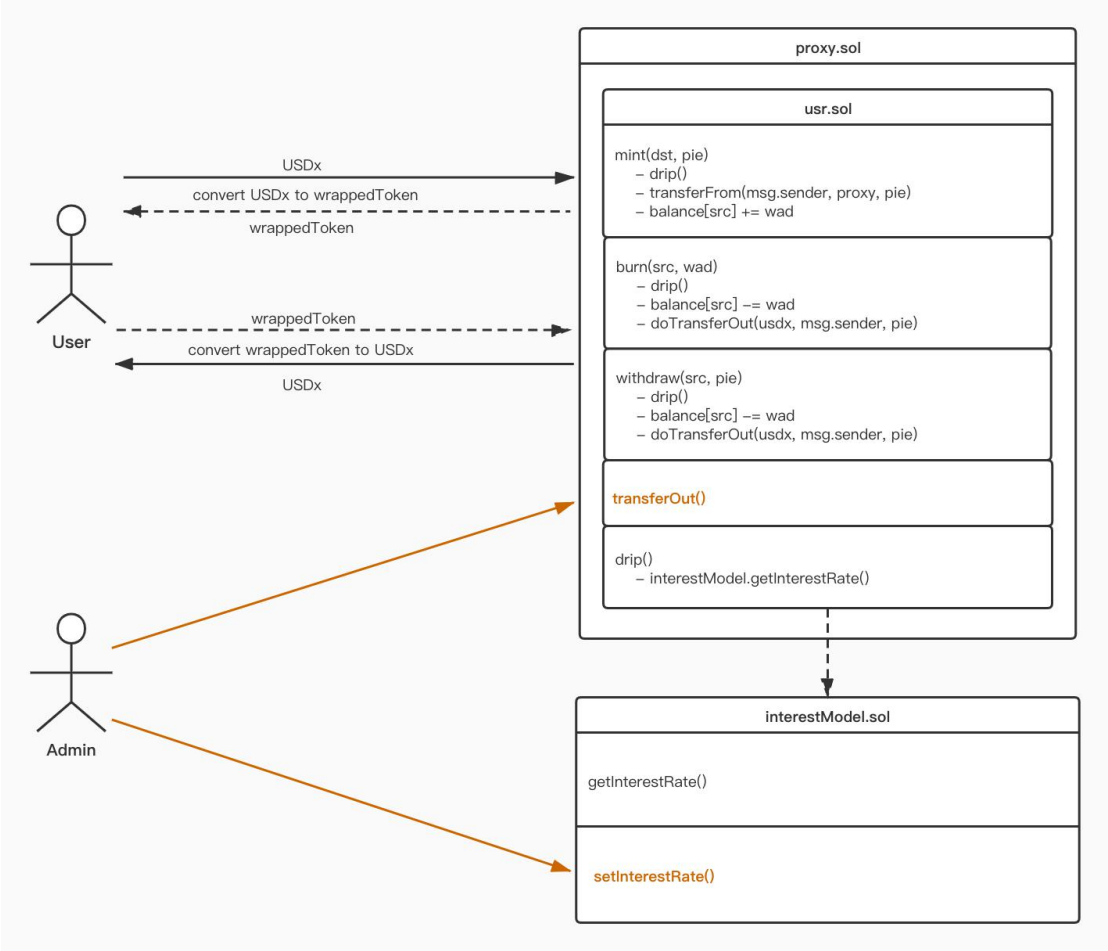
In the diagram, the USR Token and the Interest Model can be upgraded, and the arrows indicate the token circulation and exchange.

When a user deposits USDx into the USR contract, he will get some USR Token according to the exchange rate of USDx / USR Token at that time(the exchange rate is calculated according to the interest model contract, which means how much USDx a USR Token can be exchanged. For example, if you deposit 10000 USDx, when rate = 1.001, you will get 9990.01 USR Token). At the mean time, the deposited USDx will be transferred to the USR Core Proxy.

DIP-001 is an interest-generating protocol for USDx constituent stable coins. It will take the locked constituent stable coins to generate interest, and then use the interest to generate USDx again, finally transfer USDx into the USR contract.

When users want to get back USDx from the USR contract, at first, it will calculate the latest exchange rate of the USDx / USR Token at that time, then the corresponding amount of USDx is taken from the USR contract to the user, finally the user's USR Token is burned.

❖ Contracts and Interfaces



1. USR.sol

Notice：1. The contract deployer is the owner of the contract; 2. The USR contract is an ERC20 contract, so it has the basic ERC20 contract methods, including: approve(), balanceOf(), transfer(), transferFrom(), so we will not explain these functions at there.

1) mint(_dst, _pie) external:

In the case that the contract is not closed, the caller deposits '_pie' USDx for the user (_dst) to get the corresponding USR.

2) burn(_src, _wad) external:

In the case that the contract is not closed, the caller gets the corresponding USDx by burning the user (_src) '_wad' USR.

3) draw(_src, _pie) external note whenNotPaused:

In the case that the contract is not closed, the caller wants to get USDx of _pie for the user (_src).

4) getExchangeRate() public view:

Calculate the latest exchange rate of the USDx / USR Token at this moment.

5) getTotalBalance(_account) external view:

Calculate the maximum amount of USDx that the user (_account) can withdraw at this moment (principal interest minus transaction fee).

6) share() external view:

Indicates the amount of USDx that can be deposited in this contract.

7) equity() external view:

Indicates the current contract state, greater than 0 indicates a surplus, and less than 0 indicates a loss, prompting the manager to transfer more USDx to pay the interest that users can get by depositing USDx.

8) drip() public:

Updated the latest exchange rate of the USDx / USR Token.

9) takeOut(address tokenAddress, address receiver, uint256 amount) external onlyManager whenNotPaused:

In the case that the contract is not closed, the account with manager permission takes out some tokens and transfers them to receiver.

10) updateInterestModel(address _interestModel) external note onlyOwner:

The account with owner permissions replaces the address of the interest model contract.

11) setMaxDebtAmount(_newMaxDebtAmount) external onlyOwner:

The account with owner permissions updates the total number of USDx that can be deposited into USR contract.

12) updateOriginationFee(uint _newOriginationFee) external onlyOwner

The account with owner permissions resets the rate of withdrawal fees.

13) initialize(_name, _symbol, _decimals, _interestModel, _usdx, _originationFee, _maxDebtAmount):

This function is equal to the 'constructor()' function of the contract, which can be called only once, in order to cooperate with the proxy contract.

2. InterestModel.sol

Notice: The contract deployer is the owner of the contract.

1) setInterestRate(uint _interestRate) external onlyManager;

The account with manager permissions updates the value of interest rate.

2) getInterestRate() external view returns (uint);

Read the value of the interest rate.

3. Ownable.sol

Notice: 1. This is a library contract that can be inherited by many contracts; 2. The contract deployer is the owner of the contract.

1) modifier onlyOwner()

Modified functions can only be used by the owner.

2) modifier onlyManager()

Modified functions can only be used by the manager.

3) transferOwnership(address pendingOwner_) external onlyOwner:

The original owner account transfers permissions to another account (pendingOwner_).

4) acceptOwnership() external;

The new owner account (pendingOwner) calls to receive onwer permissions and become the new owner.

5) setManager(address account) external onlyOwner

The account with owner permissions sets a new account (account) with manager permissions .

6) removeManager(address account) external onlyOwner

The account with owner permissions remove the original manager account (account).

4. Pausable.sol

Notice : 1. This is a library contract that has inherited the Ownable contract, it also can be inherited by many other contracts; 2. The contract deployer is the owner of the contract.

1) pause() public whenNotPaused onlyOwner

In the case that the contract is not closed, the account with owner permission will urgently close the contract.

2) unpause() public whenPaused onlyOwner

In the case that the contract is not closed, the account with owner permission will reopen the contract.

❖ **Referencs**

1. dForce USDx
2. dForce DIP-001
3. Maker DSR
4. Chai
5. imBTC