Whitepaper: GSDI Auctions (GAUC).

Crypto Shipwright et al., 2021, v. 0.1.0

Overview

GAUC is an auction house for matching borrowers and lenders to mint new GSDIs. GSDI, or Generic Smart Debt Instruments, are a defi lending primitive backed by smart wallets containing arbitrary assets. As a primitive, GSDIs can only be accessed through dapps built on top of the GSDI primitives. GAUC is a sample dapp illustrating how to create an auctionhouse for GSDIs backed by ERC20 and ERC721 assets.

GSDIs minted through GAUC may be purchased by lending pools farming GYFI. These lending pools have strategies targeting GSDIs with certain collateral, face value, maturity dates, and prices. More details on these lending pools are in the GYFI whitepaper.

GSDI Auctions

Borrowers may use GAUC to propose new GSDI to prospective lenders as follows:

1. The borrower, in a unitary transaction, transfers ERC20/721 assets to the GZCBAuction contract along with the following parameters:
    - 1a. Price in Dai, the amount they wish to borrow against the collateral;
    - 1b. Maturity date timestamp, the time at which the GZCB will mature;
    - 1c. Maximum face value in Dai, the maximum amount the borrower is willing to repay;
    - 1d. Auction End timestamp, the time at which the auction will end.
2. Lenders bid on the Face Value of the auction. Each bid must be at least 1% lower than the previous bid with the lowest bid winning.
3. At the end of the auction, one of two things occur:
    - 3a. If there is a winning bid, the lender pays the Price plus the platform fee (if enabled) to the GAuction contract which locks the collateral, mints the GSDI, and transfers the GSDI to the winning lender.
    - 3b. If there is no winning bid, the borrower may either recreate the auction or have their assets returned.
4. As the GSDI matures, the options follow steps 6a/6b from the GSDI whitepaper

GAUC Interface

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.8.3 <0.9.0;

/// @title GSDI Auction house.
/// @author Crypto Shipwright
interface IGAUC {
    /// @custom:enum OPEN Auction is available and open to bids.
    /// @custom:enum CLAIMABLE Auction had a winning bid but has not yet had its
GSDI claimed.
    /// @custom:enum CLAIMED Auction was successfully claimed and its GSDI is still
open.
    /// @custom:enum CLOSED Auction was successfully completed and its GSDI has
been covered or seized.
    /// @custom:enum EXPIRED Time is past the auction's end timestamp with no bids.
    /// @custom:enum CANCELED The borrower canceled the auction.s
    enum AUCTION_STATUS {OPEN, CLAIMABLE, CLAIMED, CLOSED, EXPIRED, CANCELED}

    /// @notice Returns the current status of an auction.
```

```
    /// @dev The AUCTION_STATUS is calculated from its and the GSDIWallet's states.
    /// @param _auctionId Unique ID for the auction.
    /// @return auctionStatus_ Status of the auction.
    function auctionStatus(uint256 _auctionId)
        external
        view
        returns (AUCTION_STATUS auctionStatus_);

    /// @notice Returns the data for a particular auction.
    /// @param _auctionId Unique ID for the auction.
    /// @return auctionEndTimestamp_ Timestamp at which the auction closes.
    /// @return lowestBid_ Current lowest bid for the face value of the GSDI.
    /// @return maturity_ Date at which the GSDI will mature.
    /// @return price_ Price of the GSDI to be paid by the lender.
    /// @return minBidIncrement_ Min increment for bids, such as 1000 DAI.
    /// @return IGSDIWallet_ Address of the wallet to back the GSDI.
    /// @return lowestBidder_ Address of the current lowest bidder.
    /// @return borrower_ Address of the borrower who will receive the assets once
covered.
    /// @return auctionStatus_ Current status of the auction.
    function auctionInfo(uint256 _auctionId)
        external
        view
        returns (
            uint256 auctionEndTimestamp_,
            uint256 lowestBid_,
            uint256 maturity_,
            uint256 price_,
            uint256 minBidIncrement_,
            address IGSDIWallet_,
            address lowestBidder_,
            address borrower_,
            AUCTION_STATUS auctionStatus_
        );

    /// @notice Returns the balance in Dai for the user that has been deposited.
    /// @dev Includes the current locked balance. Increased on deposit; decreased
on withdraw or auction claim.
    /// @param _user User to check the balance for.
    /// @return bal_ Balance of the user.
    function balance(address _user) external view returns (uint256 bal_);

    /// @notice Returns the balance in Dai for the user that is locked in OPEN
auctions.
    /// @dev Increased whenever bidding on auctions; decreased when outbid, auction
cancel, or auction claim.
    /// @param _user User to check the balance for.
    /// @return bal_ Locked Balance of the user.
    function balanceLocked(address _user) external view returns (uint256 bal_);

    /// @notice Returns the balance in Dai available to the user to bid.
    /// @dev Equal to balance minus balanceLocked.
    /// @param _user User to check the balance for.
    /// @return bal_ Avaialable Balance of the user.
    function balanceAvailable(address _user)
        external
        view
        returns (uint256 bal_);
```

```
    /// @notice Deposits Dai for the user to use for bidding on auctions.
    /// @dev Requires approve.
    /// @param _amount Amount of Dai to transfer from the sender for the user's
account.
    /// @param _user User to credit the account for.
    function deposit(uint256 _amount, address _user) external;

    /// @notice Withdraws dai from the sender's available balance. Reverts if
greater than balanceAvailable.
    /// @param _amount Amount of Dai to withdraw.
    function withdraw(uint256 _amount) external;

    /// @notice Creates a new auction with ERC20 collateral. Transfers tokens from
the sender to the auction contract.
    /// @dev Requires approve of ERC20 token first.
    /// @param _token Token address of ERC20 collateral.
    /// @param _amount Amount of tokens to use as collateral, transferFrom the
sender.
    /// @param _borrower Borrower who will receive the collateral after the auction
is covered. Usually the sender.
    /// @param _auctionEndTimestamp Timestamp at which the auction expires.
    /// @param _maxFaceValue Maximum face value to be accepted by the buyer.
    /// @param _minBidIncrement Minimum bid delta for new bids to be accepted. For
instance, 1000 dai.
    /// @param _maturity Timestamp at which the bond matures.
    /// @param _price Amount of Dai to be sent to the borrower on purchase.
    function createERC20Auction(
        IERC20 _token,
        uint256 _amount,
        address _borrower,
        uint256 _auctionEndTimestamp,
        uint256 _maxFaceValue,
        uint256 _minBidIncrement,
        uint256 _maturity,
        uint256 _price
    ) external;

    /// @notice Creates a new auction with ERC721 collateral. Transfers tokens from
the sender to the auction contract.
    /// @dev Requires approve of ERC721 token first.
    /// @param _token Token address of ERC721 collateral.
    /// @param _ids ERC721 IDs to use as collateral, transferFrom the sender.
    /// @param _borrower Borrower who will receive the collateral after the auction
is covered. Usually the sender.
    /// @param _auctionEndTimestamp Timestamp at which the auction expires.
    /// @param _maxFaceValue Maximum face value to be accepted by the buyer.
    /// @param _minBidIncrement Minimum bid delta for new bids to be accepted. For
instance, 1000 dai.
    /// @param _maturity Timestamp at which the bond matures.
    /// @param _price Amount of Dai to be sent to the borrower on purchase.
    function createERC721Auction(
        IERC721 _token,
        uint256[] _ids,
        address _borrower,
        uint256 _auctionEndTimestamp,
        uint256 _maxFaceValue,
        uint256 _minBidIncrement,
        uint256 _maturity,
        uint256 _price
```

```
    ) external;

    /// @notice Cancels an auction. Only callable by auction's borrower on OPEN
auctions with no bids.
    /// @param _auctionId ID of the auction to cancel.
    function cancel(uint256 _auctionId) external;

    /// @notice Bids on an auction. Must be at least minBidIncrement lower than the
lowestBid.
    /// @dev Updates the lowest bid and bidder for the auction. Must be an OPEN
auction.
    /// @param _auctionId ID of the auction to bid on.
    /// @param _amount Face value bid for the proposed GSDI.
    function bid(uint256 _auctionId, uint256 _amount) external;

    /// @notice Mints a new GSDI for a CLAIMABLE auction to the winning bidder.
Callable by anyone.
    /// @param _auctionId Claimable Auction to mint the GSDI for.
    function claim(uint256 _auctionId) external;
}
```