paydece

PaydeceEscrow v3

Project Overview

Two users want to exchange goods using crypto as the payment method and they agree to use paydece decentralized escrow so that the buyer will ensure that he will receive the goods and the seller ensures that he will receive the money. Funds will remain in the smart contract until both sides validate.

Characteristics of the product

1. Users connect using their self-custodial wallet once they agree they will exchange any goods.
2. The seller creates an invoice by completing the required information and sending the invoice (link) to the buyer.
3. The buyer adds the invoice to his dashboard and creates the escrow (smart contract). Then he sends the crypto to the smart contract.
4. The smart contract escrow will hold the funds of the buyer in the middle of both wallets.
5. The seller will send the product to the buyer (he knows that the crypto are escrowed and they cannot move unless he validates). Once he transferred the goods, he validates.
6. The buyer receives the good and validates it.
7. The funds are released when both sides validate the transaction was successfully done.
8. If there is any problem during the transaction a dispute is raised, and paydece will solve it to decide the side the funds will move

Technical Information

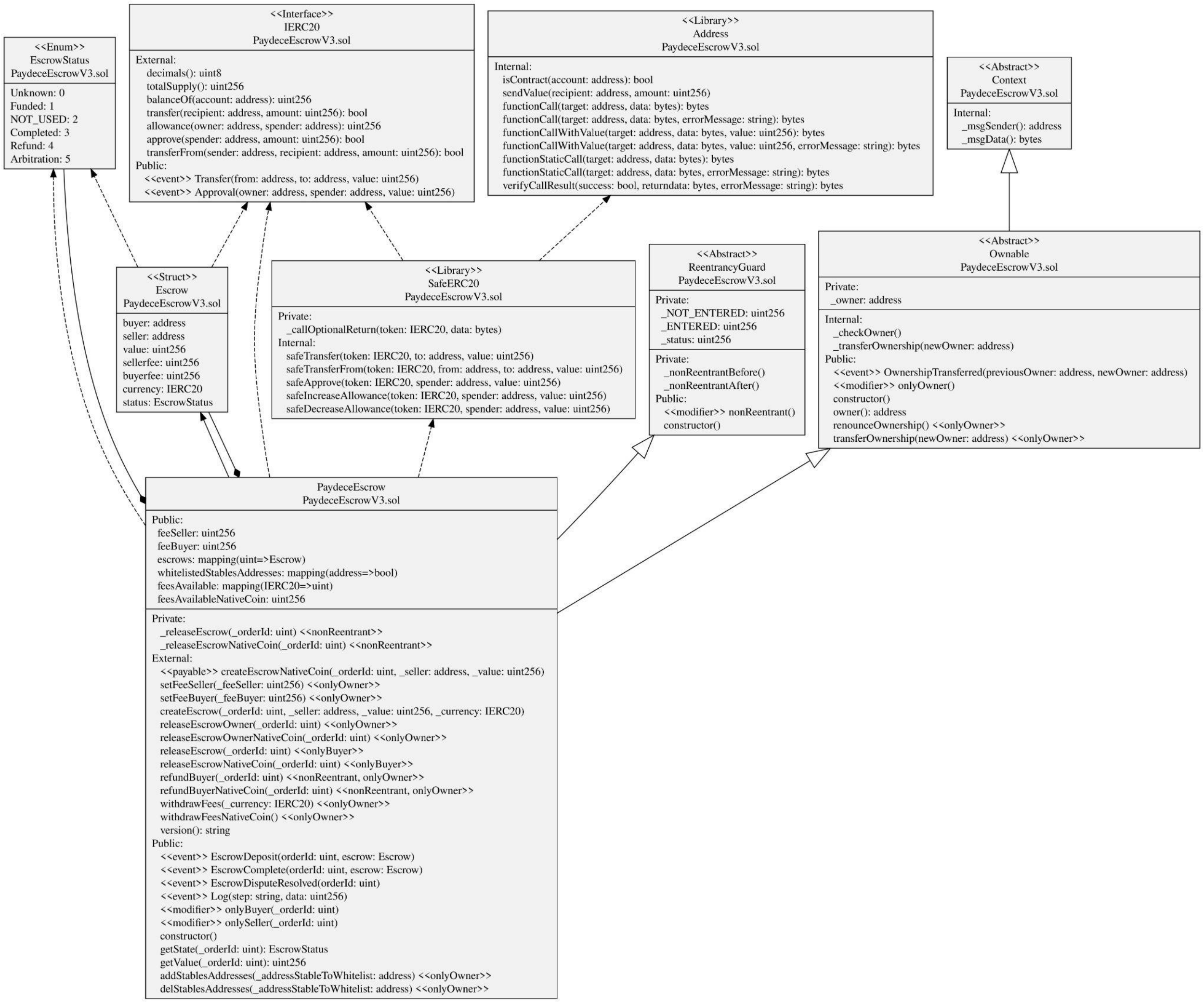
This smart contract implements an escrow service for cryptocurrency transactions. It allows the creation of an escrow for a given order, where a buyer and a seller can deposit their funds, which are held by the contract until the transaction is completed or a dispute is resolved. The contract can be used to hold funds in different currencies, as long as they are ERC20 tokens.

The contract defines a structure called Escrow to represent an escrow, which contains the addresses of the buyer and seller, the value of the transaction, the commissions for the buyer and seller, the currency used and the status of the escrow. The contract also maintains a correspondence between the order ID and the escrow to store the escrows that have been created.

The contract also has a whitelistedStablesAddresses assignment to store a list of whitelisted stablecoin addresses, and checks that the currency used in an escrow is a whitelisted stablecoin before creating the escrow. There is also a feesAvailable assignment to track fees that can be withdrawn, and a feesAvailableNativeCoin variable to track fees that can be withdrawn in the native currency (presumably ETH).

The contract has a modifier onlyBuyer and onlySeller to restrict certain functions to be called only by the escrow buyer or seller, respectively. It also has an onlyOwner modifier to restrict certain functions to be called only by the contract owner. The contract has a base contract Ownable, which provides an onlyOwner modifier and an owner address to store the address of the contract owner.

Diagram


<<Enum>>
EscrowStatus
PaydeceEscrowV3.sol
Unknown: 0
Funded: 1
NOT_USED: 2
Completed: 3
Refund: 4
Arbitration: 5
<<Interface>>
IERC20
PaydeceEscrowV3.sol
External:
decimals(): uint8
totalSupply(): uint256
balanceOf(account: address): uint256
transfer(recipient: address, amount: uint256): bool
allowance(owner: address, spender: address): uint256
approve(spender: address, amount: uint256): bool
transferFrom(sender: address, recipient: address, amount: uint256): bool
Public:
<<event>> Transfer(from: address, to: address, value: uint256)
<<event>> Approval(owner: address, spender: address, value: uint256)
<<Library>>
Address
PaydeceEscrowV3.sol
Internal:
isContract(account: address): bool
sendValue(recipient: address, amount: uint256)
functionCall(target: address, data: bytes): bytes
functionCall(target: address, data: bytes, errorMessage: string): bytes
functionCallWithValue(target: address, data: bytes, value: uint256): bytes
functionCallWithValue(target: address, data: bytes, value: uint256, errorMessage: string): bytes
functionStaticCall(target: address, data: bytes): bytes
functionStaticCall(target: address, data: bytes, errorMessage: string): bytes
verifyCallResult(success: bool, returndata: bytes, errorMessage: string): bytes
<<Abstract>>
Context
PaydeceEscrowV3.sol
Internal:
_msgSender(): address
_msgData(): bytes
<<Struct>>
Escrow
PaydeceEscrowV3.sol
buyer: address
seller: address
value: uint256
sellerfee: uint256
buyerfee: uint256
currency: IERC20
status: EscrowStatus
<<Library>>
SafeERC20
PaydeceEscrowV3.sol
Private:
_callOptionalReturn(token: IERC20, data: bytes)
Internal:
safeTransfer(token: IERC20, to: address, value: uint256)
safeTransferFrom(token: IERC20, from: address, to: address, value: uint256)
safeApprove(token: IERC20, spender: address, value: uint256)
safeIncreaseAllowance(token: IERC20, spender: address, value: uint256)
safeDecreaseAllowance(token: IERC20, spender: address, value: uint256)
<<Abstract>>
ReentrancyGuard
PaydeceEscrowV3.sol
Private:
_NOT_ENTERED: uint256
_ENTERED: uint256
_status: uint256
Private:
_nonReentrantBefore()
_nonReentrantAfter()
Public:
<<modifier>> nonReentrant()
constructor()
<<Abstract>>
Ownable
PaydeceEscrowV3.sol
Private:
_owner: address
Internal:
_checkOwner()
_transferOwnership(newOwner: address)
Public:
<<event>> OwnershipTransferred(previousOwner: address, newOwner: address)
<<modifier>> onlyOwner()
constructor()
owner(): address
renounceOwnership() <<onlyOwner>>
transferOwnership(newOwner: address) <<onlyOwner>>
PaydeceEscrow
PaydeceEscrowV3.sol
Public:
feeSeller: uint256
feeBuyer: uint256
escrows: mapping(uint=>Escrow)
whitelistedStablesAddresses: mapping(address=>bool)
feesAvailable: mapping(IERC20=>uint)
feesAvailableNativeCoin: uint256
Private:
_releaseEscrow(_orderId: uint) <<nonReentrant>>
_releaseEscrowNativeCoin(_orderId: uint) <<nonReentrant>>
External:
<<payable>> createEscrowNativeCoin(_orderId: uint, _seller: address, _value: uint256)
setFeeSeller(_feeSeller: uint256) <<onlyOwner>>
setFeeBuyer(_feeBuyer: uint256) <<onlyOwner>>
createEscrow(_orderId: uint, _seller: address, _value: uint256, _currency: IERC20)
releaseEscrowOwner(_orderId: uint) <<onlyOwner>>
releaseEscrowOwnerNativeCoin(_orderId: uint) <<onlyOwner>>
releaseEscrow(_orderId: uint) <<onlyBuyer>>
releaseEscrowNativeCoin(_orderId: uint) <<onlyBuyer>>
refundBuyer(_orderId: uint) <<nonReentrant, onlyOwner>>
refundBuyerNativeCoin(_orderId: uint) <<nonReentrant, onlyOwner>>
withdrawFees(_currency: IERC20) <<onlyOwner>>
withdrawFeesNativeCoin() <<onlyOwner>>
version(): string
Public:
<<event>> EscrowDeposit(orderId: uint, escrow: Escrow)
<<event>> EscrowComplete(orderId: uint, escrow: Escrow)
<<event>> EscrowDisputeResolved(orderId: uint)
<<event>> Log(step: string, data: uint256)
<<modifier>> onlyBuyer(_orderId: uint)
<<modifier>> onlySeller(_orderId: uint)
constructor()
getState(_orderId: uint): EscrowStatus
getValue(_orderId: uint): uint256
addStablesAddresses(_addressStableToWhitelist: address) <<onlyOwner>>
delStablesAddresses(_addressStableToWhitelist: address) <<onlyOwner>>

Functions & methods

createEscrow()

The buyer from the dApp executes this method by creating the Escrow on the Blockchain and making the transfer to put in custody the ERC20 coins.

createEscrowNativeCoin()

The buyer from the dApp executes this method by creating the Escrow on the Blockchain and making the native currency transfer to put in custody.

releaseEscrow()

The buyer from the dApp executes this method to release the coins to the seller once all steps off the Blockchain have been completed.

releaseEscrowOwner()

Paydece executes this method, with multi-signature execution, to release coins in escrow after resolving a dispute.

releaseEscrowNativeCoin()

The buyer from the dApp executes this method to release the native currency to the seller once all the steps outside the Blockchain have been performed.

releaseEscrowOwnerNativeCoin()

Paydece executes this method, with multi-signature execution, to release the native currency in escrow after resolving a dispute.

refundBuyer()

Paydece executes this method, with multi-signature execution, to return the coins in escrow to the buyer after resolving a dispute.

refundBuyerNativeCoin()

Paydece executes this method, with multi-signature execution, to return the native currency in escrow after resolving a dispute.

withdrawFees()

Paydece executes this method, with multi-signature execution, to withdraw the commissions obtained from the transactions in each currency.

withdrawFeesNativeCoin()

Paydece runs this method, with multi-signature execution, to withdraw commissions earned from transactions in the Blockchain's native currency.

5

Events

EscrowDeposit()
The event is issued when the Escrow is created and the deposit is made in custody.

EscrowComplete()
The event is issued when the Escrow is completed and the coins in custodial are released.

EscrowDisputeResolved()
The event is emitted when a dispute is resolved.