requestable-cryptokitties

Release 0.0.1

Onther Inc.

Apr 19, 2019

CONTENTS:

CHAPTER

ONE

NON REQUESTABLE CONTRACTS

Table Of Contents

1.1 General

1.1.1 ERC721

Interface for contracts conforming to ERC-721: Non-Fungible Tokens

Events

Transfer(address from, address to, uint256 tokenId)

Parameters

- `from <address >`

- `to <address >`
- `tokenId <uint256 >`

Approval(address owner, address approved, uint256 tokenId)

Parameters

- `owner <address >`
- `approved <address >`
- `tokenId <uint256 >`

External Functions

balanceOf(address _owner) public view returns (uint256 count)

Parameters

- `_owner <address>`

Returns

- `count <uint256>`

transfer(address _to, uint256 _tokenId) external

Parameters

- `_to <address>`
- `_tokenId <uint256>`

Returns

None

approve(address _to, uint256 _tokenId) external

Parameters

- `_to <address>`
- `_tokenId <uint256>`

Returns

None

transferFrom(address _from, address _to, uint256 _tokenId) external

Parameters

- `_from <address>`
- `_to <address>`
- `_tokenId <uint256>`

Returns

None

totalSupply() public view returns (uint)

Parameters

None

Returns

- `<uint256>`

ownerOf(uint256 _tokenId) external view returns (address owner)

Parameters

- `_tokenId <uint256>`

Returns

- `owner <address>`

supportsInterface(bytes4 _interfaceID) external view returns (bool)

Parameters

- `_interfaceID <bytes4>`

Returns

- `<bool>`

1.1.2 ERC721Metadata

The external contract that is responsible for generating metadata for the kitties, it has one function that will return the data as bytes.

External Functions

getMetadata(uint256 _tokenId, string) public view returns (bytes32[4] buffer, uint256 count)

Given a token Id, returns a byte array that is supposed to be converted into string.

Parameters

- `_tokenId <uint256>`
- `<string>`

Returns

- `buffer <bytes32[4]>`
- `count <uint256>`

1.2 Auctions

1.2.1 ClockAuctionBase

Contains models, variables, and internal methods for the auction.

Struct

Auction

Represents an auction on an NFT

Members

- `seller <address>`: Current owner of NFT
- `startingPrice <uint128>`: Price (in wei) at beginning of auction
- `endingPrice <uint128>`: Price (in wei) at end of auction
- `duration <uint64>`: Duration (in seconds) of auction

- `startedAt <uint64>`: Time when auction started

Events

AuctionCreated(uint256 tokenId, uint256 startingPrice, uint256 endingPrice, uint256 duration)

Parameters

- `tokenId <uint256>`
- `startingPrice <uint256>`
- `endingPrice <uint256>`
- `duration <uint256>`

AuctionSuccessful(uint256 tokenId, uint256 totalPrice, address winner)

Parameters

- `tokenId <uint256>`
- `totalPrice <uint256>`
- `winner <address>`

AuctionCancelled(uint256 tokenId)

Parameters

- `tokenId <uint256>`

1.2.2 ClockAuction

Clock auction for non-fungible tokens.

External Functions

ClockAuction(address _nftAddress, uint256 _cut)

Constructor creates a reference to the NFT ownership contract and verifies the owner cut is in the valid range.

Parameters

- `_nftAddress <address>`: address of a deployed contract implementing the Nonfungible Interface.
- `_cut <uint256>`: percent cut the owner takes on each auction, must be between 0-10,000.

Returns

None

withdrawBalance()

Remove all Ether from the contract, which is the owner's cuts as well as any Ether sent directly to the contract address. Always transfers to the NFT contract, but can be called either by the owner or the NFT contract.

Parameters

None

Returns

None

createAuction(uint256 _tokenId,uint256 _startingPrice,uint256 _endingPrice,uint256 _duration,address _seller)

Creates and begins a new auction.

Parameters

- `_tokenId <uint256>`: ID of token to auction, sender must be owner.
- `_startingPrice <uint256>`: Price of item (in wei) at beginning of auction.
- `_endingPrice <uint256>`: Price of item (in wei) at end of auction.
- `_duration <uint256>`: Length of time to move between starting price and ending price (in seconds).
- `_seller <address>`: Seller, if not the message sender

Returns

None

bid(uint256 _tokenId) payable

Bids on an open auction, completing the auction and transferring ownership of the NFT if enough Ether is supplied.

Parameters

- `_tokenId <uint256>`: ID of token to bid on.

Returns

None

cancelAuction(uint256 _tokenId)

Cancels an auction that hasn't been won yet. Returns the NFT to original owner.

Parameters

- `_tokenId <uint256>`: ID of token on auction.

Returns

None

cancelAuctionWhenPaused(uint256 _tokenId)

Cancels an auction when the contract is paused. Only the owner may do this, and NFTs are returned to the seller. This should only be used in emergencies.

Parameters

- `_tokenId <uint256>`: ID of the NFT on auction to cancel.

Returns

None

getAuction(uint256 _tokenId) view returns (address seller, uint256 startingPrice, uint256 endingPrice, uint256 duration, uint256 startedAt)

Returns auction info for an NFT on auction.

Parameters

- `_tokenId <uint256>`: ID of the NFT on auction.

Returns

- `seller <address>`
- `startingPrice <uint256>`
- `endingPrice <uint256>`

- `duration <uint256>`
- `startedAt <uint256>`

getCurrentPrice(uint256 _tokenId) view returns (uint256)

Returns the current price of an auction.

Parameters

- `_tokenId <uint256>`: ID of the token price we are checking.

Returns

- `<uint256>`

1.2.3 SaleClockAuction is ClockAuction

Clock auction modified for sale of kitties We omit a fallback function to prevent accidental sends to this contract.

Public State Variables

- `isSaleClockAuction <bool>`: Sanity check that allows us to ensure that we are pointing to the right auction in our setSaleAuctionAddress() call.
- `gen0SaleCount <uint256>`: Tracks last 5 sale price of gen0 kitty sales
- `lastGen0SalePrices <uint256[5]>`: Tracks last 5 sale price of gen0 kitty sales

External Functions

SaleClockAuction(address _nftAddr, uint256 _cut) public ClockAuction(_nftAddr, _cut)

Delegate constructor

Parameters

- `_nftAddr <address>`
- `_cut <uint256>`

Returns

None

createAuction(uint256 _tokenId, uint256 _startingPrice, uint256 _endingPrice, uint256 _duration, address _seller) external

Creates and begins a new auction.

Parameters

- `_tokenId <uint256>`: ID of token to auction, sender must be owner.
- `_startingPrice <uint256>`: Price of item (in wei) at beginning of auction.
- `_endingPrice <uint256>`: Price of item (in wei) at end of auction.
- `_duration <uint256>`: Length of auction (in seconds).
- `_seller <address>`: Seller, if not the message sender

Returns

None

bid(uint256 _tokenId) external payable

Updates lastSalePrice if seller is the nft contract Otherwise, works the same as default bid method.

Parameters

- `_tokenId <uint256>`

Returns

None

averageGen0SalePrice() external view returns (uint256)

Parameters

None

Returns

- `<uint256>`

1.2.4 SiringClockAuction is ClockAuction

Reverse auction modified for siring. We omit a fallback function to prevent accidental sends to this contract.

Public State Variables

- `isSiringClockAuction <bool>`: Sanity check that allows us to ensure that we are pointing to the right auction in our setSiringAuctionAddress() call.

External Functions

SiringClockAuction(address _nftAddr, uint256 _cut) public ClockAuction(_nftAddr, _cut)

Delegate constructor

Parameters

- `_nftAddr <address>`
- `_cut <uint256>`

Returns

None

createAuction(uint256 _tokenId, uint256 _startingPrice, uint256 _endingPrice, uint256 _duration, address _seller) external

Creates and begins a new auction. Since this function is wrapped, require sender to be KittyCore contract.

Parameters

- `_tokenId <uint256>`: ID of token to auction, sender must be owner.
- `_startingPrice <uint256>`: Price of item (in wei) at beginning of auction.
- `_endingPrice <uint256>`: Price of item (in wei) at end of auction.
- `_duration <uint256>`: Length of auction (in seconds).
- `_seller <address>`: Seller, if not the message sender

Returns

None

bid(uint256 _tokenId) external payable

Places a bid for siring. Requires the sender is the KittyCore contract because all bid methods should be wrapped. Also returns the kitty to the seller rather than the winner.

Parameters

- `_tokenId <uint256>`

Returns

None

1.3 Kitties

1.3.1 KittyAccessControl is Pausable

A facet of KittyCore that manages special access privileges.

Public State Variables

- `ceoAddress <address>`
- `cfoAddress <address>`
- `cooAddress <address>`

External Functions

setCEO(address _newCEO) external onlyCEO

Assigns a new address to act as the CEO. Only available to the current CEO.

Parameters

- `_newCEO <address>`: The address of the new CEO

Returns

None

setCFO(address _newCFO) external onlyCEO

Assigns a new address to act as the CFO. Only available to the current CEO.

Parameters

- `_newCFO <address>`: The address of the new CFO

Returns

None

setCOO(address _newCOO) external onlyCEO

Assigns a new address to act as the COO. Only available to the current CEO.

Parameters

- `_newCOO <address>`: The address of the new COO

Returns

None

1.3.2 KittyBase is KittyAccessControl

Base contract for CryptoKitties. Holds all common structs, events and base variables.

breeding kittyId sequeutnal increment . Requetable

Struct

Kitty

The main Kitty struct. Every cat in CryptoKitties is represented by a copy of this structure, so great care was taken to ensure that it fits neatly into exactly two 256-bit words. Note that the order of the members in this structure is important because of the byte-packing rules used by Ethereum. Ref: http://solidity.readthedocs.io/en/develop/miscellaneous.html

Members

- `genes <uint256>`: The Kitty's genetic code is packed into these 256-bits, the format is sooper-sekret! A cat's genes never change.
- `birthTime <uint64>`: birthTime
- `cooldownEndBlock <uint64>`: The minimum timestamp after which this cat can engage in breeding activities again. This same timestamp is used for the pregnancy timer (for matrons) as well as the siring cooldown.
- `matronId <uint32>`: The ID of the parents of this kitty, set to 0 for gen0 cats. Note that using 32-bit unsigned integers limits us to a "mere" 4 billion cats. This number might seem small until you realize that Ethereum currently has a limit of about 500 million transactions per year! So, this definitely won't be a problem for several years (even as Ethereum learns to scale).
- `sireId <uint32>`: Same as `matronId`
- `siringWithId <uint32>`: Set to the ID of the sire cat for matrons that are pregnant, zero otherwise. A non-zero value here is how we know a cat is pregnant. Used to retrieve the genetic material for the new kitten when the birth transpires.

- `cooldownIndex <uint16>`: Set to the index in the cooldown array (see below) that represents the current cooldown duration for this Kitty. This starts at zero for gen0 cats, and is initialized to floor(generation/2) for others. Incremented by one for each successful breeding action, regardless of whether this cat is acting as matron or sire.
- `generation <uint16>`: The "generation number" of this cat. Cats minted by the CK contract for sale are called "gen0" and have a generation number of 0. The generation number of all other cats is the larger of the two generation numbers of their parents, plus one.

i.e. max(matron.generation, sire.generation) + 1)

Public State Variables

- `cooldowns <uint32[14]>`: A lookup table indicating the cooldown duration after any successful breeding action, called "pregnancy time" for matrons and "siring cooldown" for sires. Designed such that the cooldown roughly doubles each time a cat is bred, encouraging owners not to just keep breeding the same cat over and over again. Caps out at one week (a cat can breed an unbounded number of times, and the maximum cooldown is always seven days).
- `secondsPerBlock <uint256>`: An approximation of currently how many seconds are in between blocks.
- `kitties <Kitty[]>`: An array containing the Kitty struct for all Kitties in existence. The ID of each cat is actually an index into this array. Note that ID 0 is a negacat, the unKitty, the mythical beast that is the parent of all gen0 cats. A bizarre creature that is both matron and sire… to itself! Has an invalid genetic code. In other words, cat ID 0 is invalid… ;-)
- `kittyIndexToOwner <mapping (uint256 => address)>`: A mapping from cat IDs to the address that owns them. All cats have some valid owner address, even gen0 cats are created with a non-zero owner.
- `ownershipTokenCount <mapping (address => uint256)>`: A mapping from owner address to count of tokens that address owns. Used internally inside balanceOf() to resolve ownership count.
- `kittyIndexToApproved <mapping (uint256 => address)>`: A mapping from KittyIDs to an address that has been approved to call transferFrom(). Each Kitty can only have one approved address for transfer at any time. A zero value means no approval is outstanding.
- `sireAllowedToAddress <mapping (uint256 => address)>`: A mapping from KittyIDs to an address that has been approved to use this Kitty for siring via breedWith(). Each Kitty can only have one approved address for siring at any time. A zero value means no approval is outstanding.
- `saleAuction <SaleClockAuction>`: The address of the ClockAuction contract that handles sales of Kitties. This same contract handles both peer-to-peer sales as well as the gen0 sales which are initiated every 15 minutes.
- `siringAuction <SiringClockAuction>`: The address of a custom ClockAuction subclassed contract that handles siring auctions. Needs to be separate from saleAuction because the actions taken on success after a sales and siring auction are quite different.

Events

Birth(address owner, uint256 kittyId, uint256 matronId, uint256 sireId, uint256 genes)

The Birth event is fired whenever a new kitten comes into existence. This obviously includes any time a cat is created through the giveBirth method, but it is also called when a new gen0 cat is created.

Parameters

- `owner <address>`
- `kittyId <uint256>`
- `matronId <uint256>`
- `sireId <uint256>`
- `genes <uint256>`

Transfer(address from, address to, uint256 tokenId)

Transfer event as defined in current draft of ERC721. Emitted every time a kitten ownership is assigned, including births.

Parameters

- `from <address >`
- `to <address >`
- `tokenId <uint256 >`

External Functions

setSecondsPerBlock(uint256 secs) external onlyCLevel

Any C-level can fix how many seconds per blocks are currently observed.

Parameters

- `secs <uint256>`

Returns

None

1.3.3 KittyOwnership is KittyBase, ERC721

The facet of the CryptoKitties core contract that manages ownership, ERC-721 (draft) compliant.

Constants

- `name <string>`
- `symbol <string>`
- `InterfaceSignature_ERC165 <bytes4>`
- `InterfaceSignature_ERC721 <bytes4>`

Public State Variables

- `erc721Metadata <ERC721Metadata>`: The contract that will return kitty metadata

External Functions

supportsInterface(bytes4 _interfaceID) external view returns (bool)

Introspection interface as per ERC-165 (https://github.com/ethereum/EIPs/issues/165). Returns true for any standardized interfaces implemented by this contract. We implement ERC-165 (obviously!) and ERC-721.

Parameters

- `_interfaceID <bytes4>`

Returns

- `<bool>`

setMetadataAddress(address _contractAddress) public onlyCEO

Set the address of the sibling contract that tracks metadata. CEO only.

Parameters

- `_contractAddress <address>`

Returns

None

balanceOf(address _owner) public view returns (uint256 count)

Returns the number of Kitties owned by a specific address.

Parameters

- `_owner <address>`: The owner address to check.

Returns

- `count <uint256>`

transfer(address _to, uint256 _tokenId) external whenNotPaused

Transfers a Kitty to another address. If transferring to a smart contract be VERY CAREFUL to ensure that it is aware of ERC-721 (or CryptoKitties specifically) or your Kitty may be lost forever. Seriously.

Parameters

- `_to <address>`: The address of the recipient, can be a user or contract.
- `_tokenId <uint256>`: The ID of the Kitty to transfer.

Returns

None

approve(address _to, uint256 _tokenId) external whenNotPaused

Grant another address the right to transfer a specific Kitty via transferFrom(). This is the preferred flow for transfering NFTs to contracts.

Parameters

- `_to <address>`: The address to be granted transfer approval. Pass address(0) to clear all approvals.
- `_tokenId <uint256>`: The ID of the Kitty that can be transferred if this call succeeds.

Returns

None

transferFrom(address _from, address _to, uint256 _tokenId) external whenNotPaused

Transfer a Kitty owned by another address, for which the calling address has previously been granted transfer approval by the owner.

Parameters

- `_from <address>`: The address that owns the Kitty to be transfered.
- `_to <address>`: The address that should take ownership of the Kitty. Can be any address, including the caller.
- `_tokenId <uint256>`: The ID of the Kitty to be transferred.

Returns

None

totalSupply() public view returns (uint)

Returns the total number of Kitties currently in existence.

Parameters

None

Returns

- `<uint256>`

ownerOf(uint256 _tokenId) external view returns (address owner)

Returns the address currently assigned ownership of a given Kitty.

Parameters

- `_tokenId <uint256>`

Returns

- `owner <address>`

tokensOfOwner(address _owner) external view returns(uint256[] ownerTokens)

Returns a list of all Kitty IDs assigned to an address. This method MUST NEVER be called by smart contract code. First, it's fairly expensive (it walks the entire Kitty array looking for cats belonging to owner), but it also returns a dynamic array, which is only supported for web3 calls, and not contract-to-contract calls.

Parameters

- `owner <address>`

Returns

- `ownerTokens <uint256[]>`

tokenMetadata(uint256 _tokenId, string _preferredTransport) external view returns (string infoUrl)

Returns a URI pointing to a metadata package for this token conforming to ERC-721 (https://github.com/ethereum/EIPs/issues/721)

Parameters

- `_tokenId <uint256>`: The ID number of the Kitty whose metadata should be returned.
- `_preferredTransport <string>`

Returns

- `infoUrl <string>`

1.3.4 KittyBreeding is KittyOwnership

A facet of KittyCore that manages Kitty siring, gestation, and birth.

Public State Variables

- `autoBirthFee <uint256>`: The minimum payment required to use breedWithAuto(). This fee goes towards the gas cost paid by whatever calls giveBirth(), and can be dynamically updated by the COO role as the gas price changes.
- `pregnantKitties <uint256>`: Keeps track of number of pregnant kitties.
- `geneScience <GeneScienceInterface>`: The address of the sibling contract that is used to implement the sooper-sekret genetic combination algorithm.

Events

Pregnant(address owner, uint256 matronId, uint256 sireId, uint256 cooldownEndBlock)

The Pregnant event is fired when two cats successfully breed and the pregnancy timer begins for the matron.

Parameters

- `owner <address>`
- `matronId <uint256>`
- `sireId <uint256>`
- `cooldownEndBlock <uint256>`

External Functions

setGeneScienceAddress(address _address) external onlyCEO

Update the address of the genetic contract, can only be called by the CEO.

Parameters

- `_address <address>`: An address of a GeneScience contract instance to be used from this point forward.

Returns

None

setAutoBirthFee(uint256 val) external onlyCOO

Updates the minimum payment required for calling giveBirthAuto(). Can only be called by the COO address. (This fee is used to offset the gas cost incurred by the autobirth daemon).

Parameters

- `val <uint256>`

Returns

None

approveSiring(address _addr, uint256 _sireId)

Grants approval to another user to sire with one of your Kitties.

Parameters

- `_addr <address>`: The address that will be able to sire with your Kitty. Set to address(0) to clear all siring approvals for this Kitty.
- `_sireId <uint256>`: A Kitty that you own that _addr will now be able to sire with.

Returns

None

isReadyToBreed(uint256 _kittyId) public view returns (bool)

Checks that a given kitten is able to breed (i.e. it is not pregnant or in the middle of a siring cooldown).

Parameters

- `_kittyId <uint256>`: reference the id of the kitten, any user can inquire about it

Returns

- `<bool>`

isPregnant(uint256 _kittyId) public view returns (bool)

Checks whether a kitty is currently pregnant.

Parameters

- `_kittyId <uint256>`: reference the id of the kitten, any user can inquire about it

Returns

- `<bool>`

canBreedWith(uint256 _matronId, uint256 _sireId) external view returns (bool)

Checks to see if two cats can breed together, including checks for ownership and siring approvals. Does NOT check that both cats are ready for breeding (i.e. breedWith could still fail until the cooldowns are finished).

Parameters

- `_matronId <uint256>`: The ID of the proposed matron.
- `_sireId <uint256>`: The ID of the proposed sire.

Returns

- `<bool>`

breedWithAuto(uint256 _matronId, uint256 _sireId) external payable whenNotPaused

Breed a Kitty you own (as matron) with a sire that you own, or for which you have previously been given Siring approval. Will either make your cat pregnant, or will fail entirely. Requires a pre-payment of the fee given out to the first caller of giveBirth()

Parameters

- `_matronId <uint256>`: The ID of the Kitty acting as matron (will end up pregnant if successful)
- `_sireId <uint256>`: The ID of the Kitty acting as sire (will begin its siring cooldown if successful)

Returns

None

giveBirth(uint256 _matronId) external payable whenNotPaused

Have a pregnant Kitty give birth! Looks at a given Kitty and, if pregnant and if the gestation period has passed, combines the genes of the two parents to create a new kitten. The new Kitty is assigned to the current owner of the matron. Upon successful completion, both the matron and the new kitten will be ready to breed again. Note that anyone can call this function (if they are willing to pay the gas!), but the new kitten always goes to the mother's owner.

Parameters

- `_matronId <uint256>`: A Kitty ready to give birth.

Returns

- `<uint256>`: The Kitty ID of the new kitten.

1.3.5 KittyAuction is KittyBreeding

Handles creating auctions for sale and siring of kitties. This wrapper of ReverseAuction exists only so that users can create auctions with only one transaction.

External Functions

setSaleAuctionAddress(address _address) external onlyCEO

Sets the reference to the sale auction.

Parameters

- `_address <address>`: Address of sale contract.

Returns

None

setSiringAuctionAddress(address _address) external onlyCEO

Sets the reference to the siring auction.

Parameters

- `_address <address>`: Address of siring contract.

Returns

None

function createSaleAuction(uint256 _kittyId, uint256 _startingPrice, uint256 _endingPrice, uint256 _duration) whenNotPaused

Put a kitty up for auction. Does some ownership trickery to create auctions in one tx.

Parameters

- `_kittyId <uint256>`
- `_startingPrice <uint256>`
- `_endingPrice <uint256>`
- `_duration <uint256>`

Returns

None

function createSiringAuction(uint256 _kittyId, uint256 _startingPrice, uint256 _endingPrice, uint256 _duration) whenNotPaused

Put a kitty up for auction to be sire. Performs checks to ensure the kitty can be sired, then delegates to reverse auction.

Parameters

- `_kittyId <uint256>`
- `_startingPrice <uint256>`
- `_endingPrice <uint256>`
- `_duration <uint256>`

Returns

None

bidOnSiringAuction(uint256 _sireId, uint256 _matronId) external payable whenNotPaused

Completes a siring auction by bidding. Immediately breeds the winning matron with the sire on auction.

Parameters

- `_sireId <uint256 >`: ID of the sire on auction.
- `_matronId <uint256 >`: ID of the matron owned by the bidder.

Returns

None

withdrawAuctionBalances() external onlyCLevel

Transfers the balance of the sale auction contract to the KittyCore contract. We use two-step withdrawal to prevent two transfer calls in the auction bid function.

Parameters

None

Returns

None

1.3.6 KittyMinting is KittyAuction

all functions related to creating kittens. PromoKitty, Gen0Kitty . Gen0Kitty

Constants

- `PROMO_CREATION_LIMIT <uint256>`: Limits the number of cats the contract owner can ever create.
- `GEN0_CREATION_LIMIT <uint256>`: Limits the number of cats the contract owner can ever create.
- `GEN0_STARTING_PRICE <uint256>`: Constants for gen0 auctions.
- `GEN0_AUCTION_DURATION <uint256>`: Constants for gen0 auctions.

Public State Variables

- `promoCreatedCount <uint256>`: Counts the number of cats the contract owner has created.
- `gen0CreatedCount <uint256>`: Counts the number of cats the contract owner has created.

External Functions

createPromoKitty(uint256 _genes, address _owner) external onlyCOO

we can create promo kittens, up to a limit. Only callable by COO

Parameters

- `_genes <uint256>`: the encoded genes of the kitten to be created, any value is accepted
- `_owner <address>`: the future owner of the created kittens. Default to contract COO

Returns

None

createGen0Auction(uint256 _genes) external onlyCOO

Creates a new gen0 kitty with the given genes and creates an auction for it.

Parameters

- `_genes <uint256>`

Returns

None

1.3.7 KittyCore is KittyMinting

CryptoKitties: Collectible, breedable, and oh-so-adorable cats on the Ethereum blockchain. The main CryptoKitties contract, keeps track of kittens so they don't wander around and get lost.

Public State Variables

- `newContractAddress <address>`: Set in case the core contract is broken and an upgrade is required

External Functions

setNewAddress(address _v2Address) external onlyCEO whenPaused

Used to mark the smart contract as upgraded, in case there is a serious breaking bug. This method does nothing but keep track of the new contract and emit a message indicating that the new address is set. It's up to clients of this contract to update to the new contract address in that case. (This contract will be paused indefinitely if such an upgrade takes place.)

Parameters

- `_v2Address <address>`: new address

Returns

None

getKitty(uint256 _id) external view returns (bool isGestating, bool isReady, uint256 cooldownIndex, uint256 nextActionAt, uint256 siringWithId, uint256 birthTime, uint256 matronId, uint256 sireId, uint256 generation, uint256 genes)

Returns all the relevant information about a specific kitty.

Parameters

- `_id <uint256>`: The ID of the kitty of interest.

Returns

- `isGestating <bool>`
- `isReady <bool>`
- `cooldownIndex <uint256>`
- `nextActionAt <uint256>`
- `siringWithId <uint256>`
- `birthTime <uint256>`
- `matronId <uint256>`
- `sireId <uint256>`
- `generation <uint256>`
- `genes <uint256>`

unpause() public onlyCEO whenPaused

Override unpause so it requires all external contract addresses to be set before contract can be unpaused. Also, we can't have newContractAddress set either, because then the contract was upgraded.

Parameters

None

Returns

None

withdrawBalance() external onlyCFO

Allows the CFO to capture the balance available to the contract. Auction Fee breedWithAuto() ETH. giveBirth .

Parameters

None

Returns

None

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search