Smart contracts documentation

Here we can find important info for the Token Dev and know more about functions inside the smart contract.

There is currently 3 Smart Contract, all coded with Solidity.

JPEGVaultToken.sol : this is the smart contract used to create the token. It includes the tax process and a function that will make a snapshot of all holders with their balance when called. This is usefull for the Redistribution.sol contract since this one needs to get all the holders and their balance to redistribute accordingly. The tax is set to 10%.

> 1% goes to the growth wallet (used to make the project grow. Same wallet as the owner, can be changed with the appropriate function, see below).
>
> 7% goes to the vault wallet (used to buy NFTs)
>
> 2% goes to the liquidity on Uniswap V2
>
> The tax can be changed anytime. Higher or lower. Each tax has a function that can be called to deal with each tax on its own.

Redistribution.sol : this is the smart contract used to create the redistribution process. It takes a snapshot of every holders and their balance to ensure that they get the % of the benefits they deserve when claiming.

JPEGVaultICO.sol : this is the smart contract used to raise funds. It works with a React UI and had a hard cap of 20 ETH. 0.05 min amount of buying and 1 ETH max amount of buying.

Smart contract addresses on mainnet :

JPEGVaultToken.sol : 0xf4d294931B7F22Ba5A9f3Ed932E6CAfc21805E48

Redistribution.sol : 0x33B0c076E27c4C273a2cfa5f683591f2C101F913

JPEGVaultICO.sol : 0xA51BBb420552572d6D54f211CC54B951581b147A

ETH/JPEG UniSwap Pair V2 : to come

JPEGVaultToken.sol functions :

Here is all of the available function in the smart contract of the token that can be used from Remix IDE or from a React UI (currently there is none).

createRedistribution : this function is called by the Redistribution.sol contract to generate a snapshot of every holder with their balance at the time of calling. It's needed to ensure that every claiming process will benefit to the holder at the time of the snapshot.

excludedRedistributionAddress : with this function we can add Address that will not benefit from the redistribution process.

removeRedistributionAddress : with this function we can remove an address that is excluded from the redistribution process, the address will then benefit from the redistribution process.

excludeTaxAddress : with this function we can exclude an address frome the tax system of the token, tax will not apply for the given address.

removeTaxAddress : witht this function we can remove an address taht is excluded from the tax, this way, the tax will reapply on the address.

setDevAddress : with this function we can set a different address to receive the 1% tax that is allocated to the growth of the project. By default, the owner address is also the dev address.

setDevFees : with this function we can change the dev fees percentage that is applied on every transaction. At the moment this value is set on 1%. We can set it to 0 or more.

setLiquidityFees : with this function we can change the liquidity fees percentage that is applied on every transaction. The 2% liquidity fee goes directly to the UniSwap V2 liquidity pool.

setMinAmountForSwap : with this function we can set a higher or lower minimum amount for the swap process to take place on the liquidity. By default this value is set to 100. This means below this value the smart contract will store the token and proceed swaping when the balance is equal or greater than 100. Function is called on each transaction. A crowd task could be useful but not mandatory since it's a very low value.

setRedistributionContract : with this function we can set the Redistribution.sol contract address. This function is only called after the deployement of the smart contract Redistribution.sol and will likely not be used anymore.

setUniswapV2Pair : with this function we can set the address of the UnisSwap v2 ETH/JPEG pair so the swapping process and the liquidity tax can be sent. This function is used once when the liquidity pool is created and will likely not be used anymore.

setVaultAddress : with this function you can set the Vault address, the vault address is receiving the 7% of the tax for the purpose of buying NFT. When the token is deployed, this address is set. But it can be changed later if needed.

setVaultFees : with this function you cans change the vault tax fee. Currently it's set to 7%.

setWETHAddress : with this function we can set the WETH smart contract address, it's likely to be used if the WETH smart contract address wasn't changed at the time of deployement inside the code, or if one day it changes. It's needed because we need to swap wETH for the liquidity pool.

Bonus : if we want to change the 10% tax of the project to 0% all that needs to be done is to set every value of those function to 0 :

setDevFees

setLiquidityFees

setVaultFees

But we can also change each taxes on it's own to increase or decrease the value.

Redistribution.sol functions :

Here is all of the available function in the smart contract of the redistribution that can be used from Remix IDE or from a React UI (currently there is none).

The purpose of this contract is to make a snapshot of every holders and their balance at the time of the redistribution so they can claim their benefits.

First of all, we need to create a redistribution with the create function and pass the total amount of claimable JPEG as a parameter.

This will create the snapshot and save the value of the claiming process and give to everyone ability to claim their benefits according to the % of the token they hold at the time of snapshot.

Then all the users needs to do is claim from the dashboard (currently not developped). If he don't claim before the next snapshot, it's not an issue because he'll automatically

be able to claim all the JPEG waiting for him.

claim : this function can be called by the holder in the dashboard. It allows the holder to claim the earned JPEG token.

create : this function takes the amount of total JPEG allocation for the redistribution. It will create the snapshot and save the value of the claiming process and give to everyone ability to claim their benefits according to the % of the token they hold at the time of snapshot.

getClaimable : this function can be used by users or admin to see how many JPEG token can be claimed by an address. On the dashboard, the holder will see how many JPEG he can claim.

JPEGVaultICO.sol functions :

This contract is used for the presale of the JPEG token. Here is how it can be configured and the functions inside.

buyTokens : this function allows the buying of the token while the sale is running.

extendTime : this function allows the admin of the smart contract, usually the owner, to extend the ending time of the sale, default is set to 1 week.

withdrawTokens : this function allows the buyer to claim the token that he bought during the sale. It can be triggered at the end of the sale.

balanceOf : this function allows the calling address to see the balance of the JPEG token that they bought.

balances : samish.

closingTime : this function returns the closing time of the sale, timestamp.

fundingTarget : this function returns the funding target of the sale, the initial funding target is set when contract is deployed.

getRate : this function returns the rate, it's how many token you'll get per ether.

getToken : this function returns the token address that is linked to the contract, in our case JPEG token.

getWallet : this function returns the wallet where the ETH raised are sent to.

hasClosed : this function returns if the sale is close or not.

isFinalized : this function returns if the sale as reached the funding goal.

isOpen : this function returns if the sale is open or not.

maxDeposit : this function returns the max deposit per user, this value is set when contract is deployed.

minDeposit : this function returns the min deposit per user, this value is set when contract is deployed.

openingTime : this function returns the opening time of the sale, timestamp.

targetReached : this function returns if the target of funding was reached or not

tokenClaimed : this function can be called to see if the address has claimed token or not

weiRaised : this function shows the amount of ETH raised