Smart Contract Software Documentation

Pablo – PABLO

How to deploy the contract on the Smart Chain

How to deploy Pablo.sol (Attachment to this documentation - *Pablo.sol*)

Read this documentation carefully before deploying on the main net.

The entry point is the contract **PABLO[:413]**.

No constructor arguments are needed,

the address to deploy will become the owner with the full total supply minted to it.

Optimization needs to be enabled on deploy.

After deploying:

1.) Call **excludeFromReward()[:976]** with uniswapV2Pair before providing any liquidity

2.) Set the BNB distribution wallets.

setCharityAddress()[:1072]

setMarketingAddress()[:1084]

setEqualizerAddress()[:1096]

setControlledBurnAddress()[:1108]

3.) Call **includeInFee()[:1030]** with uniswapV2Pair after **(!)** providing liquidity for any and all fees to take place

All PABLO specific software is found inside the contract PABLO[:413].

Total supply is set as _tTotal[:428] to 1,000,000,000,000,000 and 18 decimals.

The name is set as _name[:434] to “Pablo”.

The symbol is set as _symbol[:435] to “PABLO”.

The decimals are set as _decimals[:436] to 18.

The token amount to be burned weekly is set as controlledBurnWeeklyAmount[:467] to 10,000,000,000,00

The fees are initially set to

2% _redistributionFee[:438]	- will be used as reflection for holding
2% _burnFee[:441]	- will be deducted from total supply
2% _lpFee[:444]	- will be used to re-add to the liquidity pool
2% _marketingFee[:447]	- will be converted to BNB and sent to the marketing wallet
2% _charityFee[:450]	- will be converted to BNB and sent to the charity wallet
1% _equalizerFee[:453]	- will be converted to BNB and sent to the equalizer wallet

Every fee can be set in their setter function, all fees combined cannot be higher than 25%.
The setters are

setRedistributionFee()[:1132]

setBurnFee()[:1148]

setLpFee()[:1164]

setMarketingFee()[:1181]

setCharityFee()[:1198]

setEqualizerFee()[:1215]

The maximum transfer amount is initially set to 100% of the total supply as **_maxTxAmount[:459]** and should only be changed when the asset gets listed on multiple exchanges.

The maximum transfer can be updated in **setMaxTxPercent()[:1241]** and must be more than 0.

Operational functions

excludeFromReward()[:976]

Any active trading pair such as uniswapV2Pair needs to be excluded! After deploying, the initial uniswapV2Pair needs to be "excludedFromReward" before providing any liquidity. Any "excludeFromReward" should only happen to an address that does not hold any tokens. Any includeInReward should never be called (!), this can only be considered a failsafe.

includeInFee()[:1030]

Any active trading pair such as uniswapV2Pair should be included in fees to have the tokenomics work. Include in fees needs to happen after providing liquidity to make sure no fees on token will be taken on provide. To exclude any included from fee, excludeFromFee() can be called.

includeInSuper()[:1054]

Use this function to include addresses to be excluded from any fees no matter where they are transferring to or from. To revert the super exclude call excludeFromSuper()[:1074]

setTokenAutoSwapLimit()[:1254]

This will set the limiter for when to auto convert tokens stored on the contract. It is initially set as minimumTokensBeforeSwap[:460] to 100,000,000,000

setSwapAndLiquifyEnabled()[:1264]

Enable/disable any auto conversion to BNB from tokens stored on the contract. This will have no effect on any fees taken.

setRouterAddress()[:1273]

To set a new router address. This will also create a new pair with that router.

totalReflectionFees()[:1284]

Read the total amount of reflection token collected.

totalBurn()[:1292]

Read the total amount tokens burned.

setCharityAddress()[:1072]
setMarketingAddress()[:1084]
setEqualizerAddress()[:1096]
setControlledBurnAddress()[:1108]
setRedistributionFee()[:1132]
setBurnFee()[:1148]
setLpFee()[:1164]
setMarketingFee()[:1181]
setCharityFee()[:1198]
setEqualizerFee()[:1215]

Disclaimer

Although we take security very seriously this cannot be considered an insurance against any failure or unforeseen behaviour of any live deployment. Critical Roll does not recommend that any cryptocurrency should be bought, sold, or held by anybody. This is a documentation solely for the deployment of the attached source code file written in Solidity 0.8.6 and licenced by Apache-2.0 Copyright Critical Roll. Critical Roll or any involved developer do not take any responsibility for any use of this source code or the deployed software that could come from it.