ETHOS
AUDIT

# Dextera

## Audit Report

Nov. 11, 2021

Revision 1: Nov. 18, 2021

# Contents

# Executive Summary

## Audit Details

| Project Name | Dextera |
|---|---|
| Codebase | R1: https://www.bscscan.com/address/0xccc7023ed2c8215dc7e297d1d458d3ac9300b6f9#code |
| Source Code | Dextera.sol |
| Initial Audit Date | Nov 11, 2021 |
| Revision Dates | R1: Nov. 18, 2021 |
| Methodology | Manual, Automated |

## Methodology

This audit's objectives are to evaluate:

- Security-related issues
- Code quality
- Relevant documentation
- Adherence to specifications
- Adherence to best practices

This audit examines the possibility of issues existing along the following vectors (but not limited to):

- Single & Cross-Function Reentrancy
- Front Running (Transaction Order Dependence)
- Timestamp dependence
- Integer Overflow and Underflow
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Number rounding errors
- DoS with (Unexpected) Revert
- DoS with Block Gas Limit
- Insufficient gas griefing
- Forcibly sending native currency
- Logical oversights
- Access control
- Centralization of power
- Logic-Specification Contradiction
- Functionality duplication
- Malicious token minting

The code review conducted for this audit follows the following structure:

1. Review of specifications, documentation to assess smart contract functionality
2. Manual, line-by-line review of code
3. Code's adherence to functionality as presented by documentation
4. Automated tool-driven review of smart contract functionality
5. Assess adherence to best practices
6. Provide actionable recommendations

# Contract Details

| | |
|---|---|
| Contract IDs | R1: **0xCcC7023ed2c8215dc7E297d1D458d3AC9300b6F9** |
| Network | BSC |
| Language | Solidity |
| Compiler | v0.8.4+commit.c7e474f2 |
| Verification Date | Nov. 18, 2021 |
| Contract Type | BEP-20 Token |
| Libraries | Custom |

# Token Details

| | |
|---|---|
| Contract Name | Dextera |
| Symbol | DXTA |
| Decimals | 9 |
| Total Supply | 10,000,000,000 |
| Max Tx Amount | 100,000,000 |
| Total Tx Tax % | 14 |
| Holder Reflection % | 2 |
| Liquidity Provision % | 8 |
| Marketing Reflection % | 4 |
| Liquidity Provision Trigger | 20,000,000 |

# Result Summary

Ethos' audit of the Dextera token smart contract has concluded with a **POSITIVE** result. The initial review identified a number of non-critical issues that could have been left within the final build with simple acknowledgements and transparency. However, the Dextera team has gone above and beyond to ensure that as many of the identified issues were resolved prior to launch. This is a testament to the team's dedication to the project and their desire to focus on security and best practices. The remaining report includes all issues identified in the initial review, as well as the revised status post resolution by the team.

# Risk Levels

**LOW**

The issue is informational and does not pose an immediate risk, but is relevant to security best practices.

**MEDIUM**

The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.

**HIGH**

The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.

**EXTREME**

The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.

## Issues Reported

| Severity | Unresolved | Acknowledged | Resolved |
|---|---|---|---|
| Extreme | 0 | 0 | 0 |
| High | 0 | 0 | 1 |
| Medium | 0 | 0 | 2 |
| Low | 0 | 6 | 4 |

## Issues Summary

| ID | Title | Severity | Status |
|---|---|---|---|
| DXTA-0 | Owner Privilege | High | Resolved |
| DXTA-1 | Division before multiplication | Medium | Resolved |
| DXTA-2 | Use of non-standard SafeMath library | Medium | Resolved |
| DXTA-3 | Variables can be declared as ‘constant’ | Low | Resolved |
| DXTA-4 | State variable visibility is not set | Low | Resolved |
| DXTA-5 | Unhandled return value | Low | Acknowledged |
| DXTA-6 | Floating pragma | Low | Resolved |
| DXTA-7 | Missing Event emissions | Low | Acknowledged |
| DXTA-8 | Missing zero-address validation | Low | Resolved |
| DXTA-9 | Environment & function/variable naming mismatch | Low | Acknowledged |
| DXTA-10 | Dead code | Low | Acknowledged |
| DXTA-11 | Function initializing state | Low | Acknowledged |
| DXTA-12 | Functions that could be declared external | Low | Acknowledged |

# Detailed Findings

## DXTA-0 – Owner Privilege

| **Severity**: High | **Status**: Resolved |
|---|---|

**Description**: The "*addLiquidity*" function calls the uniswapV2Router.addLiquidityETH function with the "to" address specified as owner() for acquiring the generated LP tokens from the DXTA-BNB pool.

**Risk**: Over time the _owner address will accumulate a significant portion of LP tokens. If the _owner is an Externally Owned Account, mishandling of its private key can have devastating consequences to the project as a whole.

**Recommendation**: We advise the to address of the uniswapV2Router.addLiquidityETH function call to be replaced by the contract itself, i.e. address(this), and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the _owner account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, i.e. Multisig wallets.

Feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency
- Assignment of privileged roles to multi-sig wallets to prevent single point of failure
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

**Location**: Dextera.sol : line 743

**Team Comment**: The 'to' address has been changed to the contract address, address(this).

## DXTA-1 – Division before multiplication

| **Severity**: Medium | **Status**: Resolved |
|---|---|

**Description**: On line 685, *transferToAddressETH(marketingAddress, transferredBalance.div(_liquidityFee).mul(marketingDivisor))* division is performed before multiplication.

**Location**: Since the division can produce non-integer results with long repeating decimals, this can cause unforeseen rounding errors.

**Recommendation**: Consider ordering multiplication before division.

**Location**: Dextera.sol : line 685

## DXTA-2 – Use of non-standard SafeMath library

| **Severity**: Medium | **Status**: Resolved |
|---|---|

**Description**: The contract uses a non-standard version of the SafeMath library which may lead to possible integer overflow/underflow scenarios.

**Risk**: This can become a potentially critical scenario during variable updates which have the potential to exceed the limits of an integers upper or lower bounds. If an integer variable's value exceeds its max value during execution, the variables value will cycle back to either its min/max value, making the entire smart contract more vulnerable to attack.

**Recommendation**: It is highly recommended to use the OpenZeppelin SafeMath.sol library to mitigate the potential overflow/underflow instances.

**Location**: Dextera.sol : lines 47-99

**SWC Registry**: SWC-101

## DXTA-3 – Variables can be declared as 'constant'

| **Severity**: Low | **Status**: Resolved |
|---|---|

**Description**: Variables _tTotal , _name , _symbol and _decimals could be declared as constant since these state variables are never to be changed.

**Risk**: This is a minor gas optimization issue.

**Recommendation**: We recommend declaring these variables as 'constant' if they aren't going to be changed.

**Location**: Dextera.sol : lines 432, 443, 447, 448, 449

## DXTA-4 – State variable visibility is not set

| **Severity**: Low | **Status**: Resolved |
|---|---|

**Description**: The global state variable "*inSwapAndLiquify*" visibility is unspecified. Labelling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

**Risk**: Minor informational issue, RE: CWE-710: Improper Adherence to Coding Standards.

**Recommendation**: It is best practice to set the visibility of all state variables explicitly. The default visibility for "*inSwapAndLiquify*" is internal.

**Location**: Dextera.sol : line 467

**SWC Registry**: SWC-108

## DXTA-5 – Unhandled return value

**Severity**: Low **Status**: Acknowledged

**Description**: The return value of the function call "*addLiquidityETH*" is not checked or handled.

**Risk**: Execution will resume even if the "*addLiquidityETH*" throws an exception. If the call fails accidentally or an attacker forces the call to fail, this may cause unexpected behaviour in the subsequent program logic.

**Recommendation**: We recommend using variable to receive the return value of the "*addLiquidityETH*" function call and handle both success and failure scenarios.

**Location**: Dextera.sol : line 738

**SWC Registry**: SWC-104

## DXTA-6 – Floating pragma

**Severity**: Low **Status**: Resolved

**Description**: Contract is not set with a defined pragma solidity version, but with a floating pragma. Currently set to "*^0.8.4*".

**Risk**: Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

**Recommendation**: We recommend locking the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library.

**Location**: Dextera.sol : line 19

**SWC Registry**: SWC-103

## DXTA-7 – Missing Event emissions

**Severity**: Low **Status**: Acknowledged

**Description**: There are several functions that change state variables, however, they do not emit events to pass the changes out of chain.

**Risk**: Not emitting an event from functions that impose changes to state variables could result in a lack of functionality often required for sound logic and functionality within external applications calling on smart contract functions.

**Recommendation**: We recommend emitting events for all essential state variables that are possible to be changed during runtime.

**Location**: Dextera.sol

## DXTA-8 – Missing zero-address validation

| **Severity**: Low | **Status**: Resolved |
|---|---|

**Description**: The "*setMarketingAddress*" function does not include any validation to ensure that the address being set isn't the zero or dead address.

**Risk**: While the risk is minimal, it is still a possibility for the marketing address to be set to the zero or dead addresses, causing loss of tokens and marketing funds from the project.

**Recommendation**: We recommend adding checks to ensure that the address provided as input to functions are valid addresses for its intended use.

**Location**: Dextera.sol : line 920

## DXTA-9 – Environment & function/variable naming mismatch

| **Severity**: Low | **Status**: Acknowledged |
|---|---|

**Description**: The Dextera contract uses Pancakeswap for swapping and liquidity adds using BNB, however, functions and variable are named with Uniswap and Ethereum.

**Risk**: Mismatched function and variables names from the environment in which a smart contract operates can cause confusion.

**Recommendation**: We recommend changing Uniswap and ETH to Pancakeswap and BNB.

**Location**: Dextera.sol

## DXTA-10 – Dead code

| **Severity**: Low | **Status**: Acknowledged |
|---|---|

**Description**: The "*addLiquidity*" function is never called or used within the contract, and its visibility is private so it has no use.

**Risk**: Dead code can affect the gas cost.

**Recommendation**: We recommend removing any dead code from non-library contracts.

**Location**: Dextera.sol : line 733

**SWC Registry**: SWC-135

## DXTA-11 – Function initializing state

**Severity**: Low **Status**: Acknowledged

**Description**: A few state variables are initialized through function calls that are not pure/constant, or that use non-constant state variables

**Risk**: Users might intend a function to return a value a state variable can initialize with, without realizing the context for the contract is not fully initialized.

**Recommendation**: Remove any initialization of state variables via non-constant state variables or function calls. If variables must be set upon contract deployment, locate initialization in the constructor instead.

**Location**: Dextera.sol : lines 444, 453, 456

## DXTA-12 – Functions that could be declared external

**Severity**: Low **Status**: Acknowledged

**Description**: Several functions are declared as public visibility, however, since they are never called by the contract they should be declared external.

**Risk**: This is a gas optimization issue.

**Recommendation**: We recommend that functions that are never called by the contract to be declared as external to save gas.

**Location**: Dextera.sol

# Automated Analysis

An automated analysis was completed by running Slither on the codebase. A multitude of issues were detected, however, only the issues that were deemed to be relevant to the security of the smart contract have been shown below.

Divide before multiply

```
Dextera.swapTokens(uint256) (Desktop/Code/Dextera.sol#678-687) performs a multiplication on the
result of a division:
        -
transferToAddressETH(marketingAddress,transferredBalance.div(_liquidityFee).mul(marketingDiviso
r)) (Desktop/Code/Dextera.sol#685)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
```

Unused return

```
Dextera.addLiquidity(uint256,uint256) (Desktop/Code/Dextera.sol#733-746) ignores return value
by uniswapV2Router.addLiquidityETH{value:
ethAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp)
(Desktop/Code/Dextera.sol#738-745)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

Local variable shadowing

```
Dextera.allowance(address,address).owner (Desktop/Code/Dextera.sol#539) shadows:
        - Ownable.owner() (Desktop/Code/Dextera.sol#173-175) (function)
Dextera._approve(address,address,uint256).owner (Desktop/Code/Dextera.sol#630) shadows:
        - Ownable.owner() (Desktop/Code/Dextera.sol#173-175) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-
shadowing
```

Missing events arithmetic

```
Dextera.setTaxFeePercent(uint256) (Desktop/Code/Dextera.sol#896-898) should emit an event for:
        - _taxFee = taxFee (Desktop/Code/Dextera.sol#897)
Dextera.setLiquidityFeePercent(uint256) (Desktop/Code/Dextera.sol#900-902) should emit an event
for:
        - _liquidityFee = liquidityFee (Desktop/Code/Dextera.sol#901)
Dextera.setMaxTxAmount(uint256) (Desktop/Code/Dextera.sol#904-906) should emit an event for:
        - _maxTxAmount = maxTxAmount (Desktop/Code/Dextera.sol#905)
Dextera.setMarketingDivisor(uint256) (Desktop/Code/Dextera.sol#908-910) should emit an event
for:
        - marketingDivisor = divisor (Desktop/Code/Dextera.sol#909)
Dextera.setNumTokensSellToAddToLiquidity(uint256) (Desktop/Code/Dextera.sol#912-914) should
emit an event for:
        - minimumTokensBeforeSwap = _minimumTokensBeforeSwap (Desktop/Code/Dextera.sol#913)
Dextera.setBuybackUpperLimit(uint256) (Desktop/Code/Dextera.sol#916-918) should emit an event
for:
        - buyBackUpperLimit = buyBackLimit * 10 ** 18 (Desktop/Code/Dextera.sol#917)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-
arithmetic
```

Missing zero address validation

```
Dextera.setMarketingAddress(address)._marketingAddress (Desktop/Code/Dextera.sol#920) lacks a
zero-check on :
                - marketingAddress = address(_marketingAddress) (Desktop/Code/Dextera.sol#921)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-
validation
```

### Dead Code

```
Address._functionCallWithValue(address,bytes,uint256,string) (Desktop/Code/Dextera.sol#140-157)
is never used and should be removed
Address.functionCall(address,bytes) (Desktop/Code/Dextera.sol#123-125) is never used and should
be removed
Address.functionCall(address,bytes,string) (Desktop/Code/Dextera.sol#127-129) is never used and
should be removed
Address.functionCallWithValue(address,bytes,uint256) (Desktop/Code/Dextera.sol#131-133) is
never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (Desktop/Code/Dextera.sol#135-138)
is never used and should be removed
Address.isContract(address) (Desktop/Code/Dextera.sol#103-112) is never used and should be
removed
Address.sendValue(address,uint256) (Desktop/Code/Dextera.sol#114-120) is never used and should
be removed
Context._msgData() (Desktop/Code/Dextera.sol#26-29) is never used and should be removed
Dextera.addLiquidity(uint256,uint256) (Desktop/Code/Dextera.sol#733-746) is never used and
should be removed
SafeMath.mod(uint256,uint256) (Desktop/Code/Dextera.sol#91-93) is never used and should be
removed
SafeMath.mod(uint256,uint256,string) (Desktop/Code/Dextera.sol#95-98) is never used and should
be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

### Function initializing state

```
Dextera._rTotal (Desktop/Code/Dextera.sol#444) is set pre-construction with a non-constant
function or state variable:
       - (MAX - (MAX % _tTotal))
Dextera._previousTaxFee (Desktop/Code/Dextera.sol#453) is set pre-construction with a non-
constant function or state variable:
       - _taxFee
Dextera._previousLiquidityFee (Desktop/Code/Dextera.sol#456) is set pre-construction with a
non-constant function or state variable:
       - _liquidityFee
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#function-initializing-
state
```

### Incorrect versions of Solidity

```
Pragma version^0.8.4 (Desktop/Code/Dextera.sol#19) necessitates a version too recent to be
trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.8 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-
solidity
```

### State variables that could be declared constant

```
Dextera._decimals (Desktop/Code/Dextera.sol#449) should be constant
Dextera._name (Desktop/Code/Dextera.sol#447) should be constant
Dextera._symbol (Desktop/Code/Dextera.sol#448) should be constant
Dextera._tTotal (Desktop/Code/Dextera.sol#443) should be constant
Dextera.deadAddress (Desktop/Code/Dextera.sol#432) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-
could-be-declared-constant
```

### Public function that could be declared external

```
renounceOwnership() should be declared external:
       - Ownable.renounceOwnership() (Desktop/Code/Dextera.sol#182-185)
```

```
transferOwnership(address) should be declared external:
	- Ownable.transferOwnership(address) (Desktop/Code/Dextera.sol#187-191)
getUnlockTime() should be declared external:
	- Ownable.getUnlockTime() (Desktop/Code/Dextera.sol#193-195)
getTime() should be declared external:
	- Ownable.getTime() (Desktop/Code/Dextera.sol#197-199)
lock(uint256) should be declared external:
	- Ownable.lock(uint256) (Desktop/Code/Dextera.sol#201-206)
unlock() should be declared external:
	- Ownable.unlock() (Desktop/Code/Dextera.sol#208-213)
name() should be declared external:
	- Dextera.name() (Desktop/Code/Dextera.sol#513-515)
symbol() should be declared external:
	- Dextera.symbol() (Desktop/Code/Dextera.sol#517-519)
decimals() should be declared external:
	- Dextera.decimals() (Desktop/Code/Dextera.sol#521-523)
totalSupply() should be declared external:
	- Dextera.totalSupply() (Desktop/Code/Dextera.sol#525-527)
transfer(address,uint256) should be declared external:
	- Dextera.transfer(address,uint256) (Desktop/Code/Dextera.sol#534-537)
allowance(address,address) should be declared external:
	- Dextera.allowance(address,address) (Desktop/Code/Dextera.sol#539-541)
approve(address,uint256) should be declared external:
	- Dextera.approve(address,uint256) (Desktop/Code/Dextera.sol#543-546)
transferFrom(address,address,uint256) should be declared external:
	- Dextera.transferFrom(address,address,uint256) (Desktop/Code/Dextera.sol#548-552)
increaseAllowance(address,uint256) should be declared external:
	- Dextera.increaseAllowance(address,uint256) (Desktop/Code/Dextera.sol#554-557)
decreaseAllowance(address,uint256) should be declared external:
	- Dextera.decreaseAllowance(address,uint256) (Desktop/Code/Dextera.sol#559-562)
isExcludedFromReward(address) should be declared external:
	- Dextera.isExcludedFromReward(address) (Desktop/Code/Dextera.sol#564-566)
totalFees() should be declared external:
	- Dextera.totalFees() (Desktop/Code/Dextera.sol#568-570)
minimumTokensBeforeSwapAmount() should be declared external:
	- Dextera.minimumTokensBeforeSwapAmount() (Desktop/Code/Dextera.sol#572-574)
buyBackUpperLimitAmount() should be declared external:
	- Dextera.buyBackUpperLimitAmount() (Desktop/Code/Dextera.sol#576-578)
deliver(uint256) should be declared external:
	- Dextera.deliver(uint256) (Desktop/Code/Dextera.sol#580-587)
reflectionFromToken(uint256,bool) should be declared external:
	- Dextera.reflectionFromToken(uint256,bool) (Desktop/Code/Dextera.sol#590-599)
excludeFromReward(address) should be declared external:
	- Dextera.excludeFromReward(address) (Desktop/Code/Dextera.sol#607-615)
isExcludedFromFee(address) should be declared external:
	- Dextera.isExcludedFromFee(address) (Desktop/Code/Dextera.sol#884-886)
excludeFromFee(address) should be declared external:
	- Dextera.excludeFromFee(address) (Desktop/Code/Dextera.sol#888-890)
includeInFee(address) should be declared external:
	- Dextera.includeInFee(address) (Desktop/Code/Dextera.sol#892-894)
setBuyBackEnabled(bool) should be declared external:
	- Dextera.setBuyBackEnabled(bool) (Desktop/Code/Dextera.sol#929-932)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

## Code Documentation

The code has a moderate amount of comments. This could be improved in order to help others understand the contract.

## Adherence to Specifications

The smart contract adheres to the smart contract functionality described by the Dextera team and is in line with its intended usage.

## Adherence to Best Practices

The smart contract adheres to the majority of best practices associated with a standard BEP-20 token. The few things that don't follow the best practices are noted in the automated review.

# On-Chain Analysis

## Token/Holder Distribution

To be completed on future revisions post-launch of the project.

## Priviledged Transactions

To be completed on future revisions post-launch of the project.

## Liquidity

To be completed on future revisions post-launch of the project.

# Appendix

## Revision History

Initial Review: Nov. 11, 2021 – Audit results delivered for contract ID: 0x13AA46B576f75161e4F12e8F3d2cdaEafc9854Ac

Revision #1: Nov. 18, 2021 – Issues DXTA-0, 1, 2, 3, 4, 6, 8 resolved and redeployed contract ID: 0x2E16558a2068238F20815b29b54D2602Dd435120

## Functions

| Function | Parameters | Visibility | Modifiers | Returns | Requires | Events |
|---|---|---|---|---|---|---|
| constructor | | | | | | Transfer |
| name | | public | | string | | |
| symbol | | public | | string | | |
| decimals | | public | | uint8 | | |
| totalSupply | | public | | uint256 | | |
| balanceOf | address account | public | | uint256 | | |
| transfer | address recipient, uint256 amount | public | | bool | | |
| allowance | address owner, address spender | public | | uint256 | | |
| approve | address spender, uint256 amount | public | | bool | | |
| transferFrom | address sender, address recipient, uint256 amount | public | | bool | | |
| increaseAllowance | address spender, uint256 addedValue | public | | bool | | |
| decreaseAllowance | address spender, uint256 subtractedValue | public | | bool | | |
| isExcludedFromReward | address account | public | | bool | | |
| totalFees | | public | | uint256 | | |
| minimumTokensBeforeSwapAmount | | public | | uint256 | | |
| buyBackUpperLimitAmount | | public | | uint256 | | |
| deliver | uint256 tAmount | public | | | | |
| reflectionFromToken | uint256 tAmount, bool deductTransferFee | public | | uint256 | | |
| tokenFromReflection | uint256 rAmount | public | | uint256 | | |
| excludeFromReward | address account | public | onlyOwner | | | |
| includeInReward | address account | public | onlyOwner | | (_isExcluded[account], "Account is not excluded") | |
| _approve | address owner, address spender, uint256 amount | private | | | (owner != address(0), "ERC20: approve from the zero address") (spender != address(0), "ERC20: approve to the zero address") | Approval |

| | | | | | | |
|---|---|---|---|---|---|---|
| _transfer | address from, address to, uint256 amount | private | | | (from != address(0), "ERC20: transfer from the zero address") (to != address(0), "ERC20: transfer to the zero address") (amount > 0, "Transfer amount must be greater than zero") require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount.") | |
| swapTokens | uint256 contractTokenBalance | private | lockTheSwap | | | |
| buyBackTokens | uint256 amount | private | lockTheSwap | | | |
| swapTokensForEth | uint256 tokenAmount | private | | | | SwapTokensForETH |
| swapETHForTokens | uint256 amoun | private | | | | SwapETHForTokens |
| addLiquidity | uint256 tokenAmount, uint256 ethAmount | private | | | | |
| _tokenTransfer | address sender, address recipient, uint256 amount, bool takeFee | private | | | | |
| _transferStandard | address sender, address recipient, uint256 tAmount | private | | | | Transfer |
| _transferToExcluded | address sender, address recipient, uint256 tAmount | private | | | | Transfer |
| _transferFromExcluded | address sender, address recipient, uint256 tAmount | private | | | | Transfer |
| _transferBothExcluded | address sender, address recipient, uint256 tAmount | private | | | | Transfer |
| _reflectFee | uint256 rFee, uint256 tFee | private | | | | |
| _getValues | uint256 tAmount | private | | uint256, uint256, uint256, uint256, uint256, uint256 | | |
| _getTValues | uint256 tAmount | private | | uint256, uint256, uint256 | | |
| _getRValues | uint256 tAmount, uint256 tFee, uint256 tLiquidity, uint256 currentRate | private | | uint256, uint256, uint256 | | |
| _getRate | | private | | uint256 | | |
| _getCurrentSupply | | private | | uint256, uint256 | | |
| _takeLiquidity | uint256 tLiquidity | private | | | | |
| calculateTaxFee | uint256 _amount | private | | uint256 | | |
| calculateLiquidityFee | uint256 _amount | private | | uint256 | | |
| removeAllFee | | private | | | | |
| restoreAllFee | | private | | | | |
| isExcludedFromFee | address account | public | | bool | | |
| excludeFromFee | address account | public | onlyOwner | | | |
| includeInFee | address account | public | onlyOwner | | | |

| | | | | | |
|---|---|---|---|---|---|
| setTaxFeePercent | uint256 taxFee | external | onlyOwner | | |
| setLiquidityFeePercent | uint256 liquidityFee | external | onlyOwner | | |
| setMaxTxAmount | uint256 maxTxAmount | external | onlyOwner | | |
| setMarketingDivisor | uint256 divisor | external | onlyOwner | | |
| setNumTokensSellToAddToLiquidity | uint256 _minimumTokensBeforeSwap | external | onlyOwner | | |
| setBuybackUpperLimit | uint256 buyBackLimit | external | onlyOwner | | |
| setMarketingAddress | address marketingAddress | external | onlyOwner | _marketingAddress != address(0x0 | |
| setSwapAndLiquifyEnabled | bool _enabled | public | onlyOwner | | SwapAndLiquifyEnabledUpdated |
| setBuyBackEnabled | bool _enabled | public | onlyOwner | | BuyBackEnabledUpdated |
| prepareForPreSale | | external | onlyOwner | | |
| afterPreSale | | external | onlyOwner | | |
| transferToAddressETH | address payable recipient, uint256 amount | private | | | |

# Global Variables

| Variable | Type | Visibilty | Read by Functions | Written by Functions |
|---|---|---|---|---|
| marketingAddress | address payable | public | swapTokens | setMarketingAddress |
| deadAddress | address | public constant | swapETHForTokens | |
| _rOwned | mapping (address => uint256) | private | balanceOf, excludeFromReward, _getCurrentSupply | constructor,deliver, _transferStandard, _transferToExcluded, _transferFromExcluded, _transferBothExcluded, _takeLiquidity |
| _tOwned | mapping (address => uint256) | private | balanceOf, _getCurrentSupply | excludeFromReward, includeInReward, _transferToExcluded, _transferFromExcluded, _transferBothExcluded, _takeLiquidity |
| _allowances | mapping (address => mapping (address => uint256)) | private | allowance, transferFrom, increaseAllowance, decreaseAllowance, | _approve |
| _isExcludedFromFee | mapping (address => bool) | private | _transfer, isExcludedFromFee, excludeFromFee, includeInFee | constructor |
| _isExcluded | mapping (address => bool) | private | balanceOf, isExcludedFromReward, deliver, excludeFromReward, includeInReward, _tokenTransfer, _takeLiquidity, | excludeFromReward, includeInReward |
| _excluded | address[] | private | _getCurrentSupply, | excludeFromReward, includeInReward, |
| MAX | uint256 | private constant | | |
| _tTotal | uint256 | private constant | constructor, totalSupply, reflectionFromToken, _getCurrentSupply | |
| _rTotal | uint256 | private | constructor, tokenFromReflection, _getCurrentSupply, | deliver, _reflectFee |
| _tFeeTotal | uint256 | private | totalFees, | deliver, _reflectFee |

| | | | | |
|---|---|---|---|---|
| _name | string | private constant | name | |
| _symbol | string | private constant | symbol | |
| _decimals | uint8 | private constant | decimals | |
| _taxFee | uint256 | public | calculateTaxFee, removeAllFee, restoreAllFee | _transfer, setTaxFeePercent,prepareForPreSale, afterPreSale, setAddressFee, |
| _previousTaxFee | uint256 | private | restoreAllFee | removeAllFee |
| _liquidityFee | uint256 | public | calculateLiquidityFee, removeAllFee | _transfer, restoreAllFee, setLiquidityFeePercent, prepareForPreSale, afterPreSale, setAddressFee, |
| _previousLiquidityFee | uint256 | private | restoreAllFee | removeAllFee |
| marketingDivisor | uint256 | public | | setMarketingDivisor |
| _maxTxAmount | uint256 | public | | prepareForPreSale, afterPreSale, setMaxTxAmount |
| minimumTokensBeforeSwap | uint256 | private | minimumTokensBeforeSwapAmount, _transfer | setNumTokensSellToAddToLiquidity |
| buyBackUpperLimit | uint256 | private | _transfer, buyBackUpperLimitAmount | setBuybackUpperLimit |
| uniswapV2Router | IUniswapV2Router02 | public immutable | swapTokensForEth, swapETHForTokens, addLiquidity | constructor |
| uniswapV2Pair | address | public immutable | _transfer | constructor |
| inSwapAndLiquify | bool | | _transfer | lockTheSwap |
| swapAndLiquifyEnabled | bool | public | | |
| buyBackEnabled | bool | public | | |

# Balance Updates

| Function | Changes |
|---|---|
| constructor | _rOwned[_msgSender()] = _rTotal |
| deliver | _rOwned[sender] = _rOwned[sender].sub(rAmount) |
| _transferStandard | _rOwned[sender] = _rOwned[sender].sub(rAmount)<br>_rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) |
| _transferToExcluded | _rOwned[sender] = _rOwned[sender].sub(rAmount)<br>_rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) |
| _transferFromExcluded | _rOwned[sender] = _rOwned[sender].sub(rAmount)<br>_rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) |
| _transferBothExcluded | _rOwned[sender] = _rOwned[sender].sub(rAmount)<br>_rOwned[recipient] = _rOwned[recipient].add(rTransferAmount) |
| _takeLiquidity | _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity) |