Arcane Token contract

Summary

Title	Arcane Token
Description	ERC20 token with reflection user's balances mechanism
Solidity Version	0.8.9
License	MIT
Author	Alina Kosak
Repository	https://github.com/ArcaneDeFi/arcane-contracts/blob/dev-hardhat/contracts/ArcaneToken.sol

Contract sources

Testnet (BscScan Testnet)

Address of **TransparentUpgradeableProxy** contract - 0x31c43c8Adee697c9608FF77dE8fd408B6Ec52945

Address of proxy contract **implementation** - 0x02b6E468Ec0B7246D524062ACC880B3F12bE46E4

Use cases & Usage Scenarios

ArcaneToken

It is ArcaneToken contract what is responsible for swap and liquify token depending on the threshold, different transfer tokens for accounts with and without fee and etc.

Usage Scenarios

Name of function	Function's description
`receive()`	receive ethers when ethers is sent to a contract with no calldata
`initialize(address _router, address _owner)`	initialization of contract • `address _router` - address of router • `address _owner` - address of owner
`setThreshold(uint256 threshold)`	determine the threshold for the accumulation by the owner • `uint256 threshold` - value of the threshold
`includeInReward(address account)`	include account in reward by the owner • `address account` - user's address
`setTaxFeePercent(uint256 taxFee)`	set value of the tax fee percent by the owner • `uint256 taxFee` - value of the tax fee percent
`setLiquidityFeePercent(uint256 liquidityFee)`	set value of the liquidity fee percent by the owner • `uint256 liquidityFee` - value of the liquidity fee percent
`setMaxTxPercent(uint256 maxTxPercent)`	set value of the max tx percent with the previous calculation by the owner • `uint256 maxTxPercent` - value for max tx percent
`setRouter(address _router)`	set new address of the router by the owner • `address _router` - new address of the router
`withdrawLeftovers()`	withdraw amount that is as remainder in contract by the owner

`withdrawAlienToken(address token, address recipient, uint256 amount)`	withdraw alien tokens from the balance of the contract by the owner. Also allow to withdraw arcane tokens from the contract balance in case if `_swapAndLiquifyEnabled` is disable • `address token` - address of alien token • `address recipient` - address of account that get transfer's amount • `uint256 amount` - amount of token to transfer
`deliver(uint256 tAmount)`	set value of a few variables depending on `tAmount` • `uint256 tAmount` - value of amount for set new values for a few variables
`excludeFromReward(address account)`	exclude account from reward by the owner • `address account` - address of account
`excludeFromFee(address account)`	exclude account from fee by the owner • `address account` - address of account
`includeInFee(address account)`	include account in fee by the owner • `address account` - address of account
`isExcludedFromFee(address account)`	return info about exclude account from fee • `address account` - address of account
`setSwapAndLiquifyEnabled(bool _enabled)`	set enable for swap and liquify by the owner • `bool _enabled` - bool value for add
`getUnlockTime()`	return setted lock time
`lock(uint256 time)`	locks the contract for the owner • uint256 `time` - value for set time for lock
`unlock()`	unlocks the contract for the owner
`isExcludedFromReward(address account)`	return info about exclude account from reward • `address account` - address of user
`totalFees()`	return value of total fees
`reflectionFromToken(uint256 tAmount, bool deductTransferFee)`	return amount of reflection tokens with fee and without fee • `uint256 tAmount` - amount of tokens • `bool deductTransferFee` - value to specify which calculation need to do, with fee of without fee.
`balanceOf(address account)`	return account's balance depending on account's exclude • `address account` - account's address
`totalSupply()`	return value of variable `_tTotal` (total supply)
`decimals()`	return value of the decimals
`tokenFromReflection(uint256 rAmount)`	return amount of tokens from reflection tokens • `uint256 rAmount` - amount of reflected tokens
`transfer(address recipient, uint256 amount)`	transfer amount of tokens from sender to recipient with including taxes if some of users is included in fee • `address recipient` - address of recipient • `uint256 amount` - amount of tokens to transfer

Events

ArcaneToken

- `Threshold(uint256 threshold);`

This event is emitted when the owner sets a new value of the threshold.

Argument:

-`threshold` - new value of the threshold;

- `SwapAndLiquifyEnabledUpdated(bool enabled);`

This event is emitted when the owner sets a new value of the variable swapAndLiquifyEnabled for enable/disable swap and liquify.

Argument:

-`enabled` - new value of the swapAndLiquifyEnabled;

- `SwapAndLiquify(uint256 tokensSwapped, uint256 ethReceived, uint256 tokensIntoLiquidity);`

This event is emitted when the threshold reached and will call function swapAndLiquify where will swap tokens and liquify.

Arguments:

-`tokensSwapped` - tokens from balance of contract for swap;

-`ethReceived` - contract's balance after swap;

-`tokensIntoLiquidity` - the leftovers of tokens on the balance of contract;

- `Deliver(address indexed sender, uint256 rAmount, uint256 rTotal, uint256 tFeeTotal);`

This event is emitted when user calls function deliver() and sets value for a few variables depending on amount that added user.

Arguments:

-`sender` - address of user that calls function;

-`rAmount` - value got as result a calculation from function `_getRValues()`;

-`rTotal` - value got as result a calculation from function `_getRValues()`;

-`tFeeTotal` - value of private variable `_tFeeTotal` as result a calculation;

- `ExcludeFromReward(address indexed account, uint256 tOwned);`

This event is emitted when user's account will exclude from rewards.

Arguments:

-`account` - address of account that is excluded;

-`tOwned` - value of tokens from reflections;

- `IncludeInReward(address indexed account, uint256 tOwned);`

This event is emitted when user's account will include in rewards.

Arguments:

-`account` - address of account that is included;

-`tOwned` - value of tokens from reflections (should be zero's value);

- `TransferFromSender(address indexed sender, uint256 tOwned, uint256 rOwned);`

This event is emitted when called function `_transferBothExcluded` *and sender with recipient are excluded from rewards.*

Arguments:

-`sender`- address of account that transfer amount;

-`tOwned` - value of tokens from reflections for sender;

-`rOwned` - value of reflection from tokens for sender;

- `TransferToRecipient(address indexed recipient, uint256 tOwned, uint256 rOwned);`

This event is emitted when called function `_transferBothExcluded` and sender with recipient are excluded from rewards.

Arguments:

-`recipient`- address of account that get transfer’s amount;

-`tOwned` - value of tokens from reflections for recipient;

-`rOwned` - value of reflection from tokens for recipient;

- `ExcludeFromFee(address indexed account, bool isExcludedFromFee);`

This event is emitted when account’s address will exclude from fee.

Arguments:

-`account`- address of account that is excluded;

-`isExcludedFromFee`- boolean value about include/exclude (should be true);

- `IncludeInFee(address indexed account, bool isExcludedFromFee);`

This event is emitted when account’s address will include in fee.

Arguments:

-`account`- address of account that is included;

-`isExcludedFromFee`- boolean value about include/exclude (should be false);

- `TaxFeePercent(uint256 taxFee);`

This event is emitted when owner set new value of tax fee.

Argument:

-`taxFee`- new value of tax fee;

- `LiquidityFeePercent(uint256 liquidityFee);`

This event is emitted when owner set new value of liquidity fee.

Argument:

-`liquidityFee`- new value of liquidity fee;

- `MaxTxPercent(uint256 maxTxAmount);`

This event is emitted when owner set new value for calculation value of max tx amount.

Argument:

-`maxTxAmount`- value got as result from calculation max tx amount;

- `ReflectFee(uint256 rTotal, uint256 tFeeTotal);`

This event is emitted when will transfer tokens and call function _reflectFee.

Arguments:

-`rTotal`- new value `_rTotal` got as result from calculation;

-`tFeeTotal`- new value `_tFeeTotal` got as result from calculation;

- `TakeLiquidity(uint256 rOwned, uint256 tOwned);`

This event is emitted when user will transfer tokens and call function _takeLiquidity.

Arguments:

-`rOwned`- new value of reflection from tokens for arcane’s token;

-`tOwned`- new value of tokens from reflections for arcane’s token (if address isn’t excluded from rewards then value should be zero);

- `RemoveAllFee(uint256 previousTaxFee, uint256 previousLiquidityFee, uint256 taxFee, uint256 liquidityFee);`

This event is emitted when user will transfer tokens without fee and if general set fee isn’t zero.

Arguments:

-`previousTaxFee`- value of previous tax fee;

-`previousLiquidityFee`- value of previous liquidity fee;

-`taxFee`- new value of tax fee (should be zero);

-`liquidityFee`- new value of liquidity fee (should be zero);

- `RestoreAllFee(uint256 taxFee, uint256 liquidityFee);`

This event is emitted when user will transfer tokens without fee.

Arguments:

-`taxFee`- value of previous tax fee;

-`liquidityFee`- value of previous liquidity fee;

- `TransferStandard(address indexed sender, address indexed recipient, uint256 rOwnedSender, uint256 rOwnedRecipient);`

This event is emitted when user will standard transfer tokens.

Arguments:

-`sender`- address of account that transfer amount;

-`recipient`- address of account that get transfer's amount;

-`rOwnedSender`- value of reflection from tokens for sender;

-`rOwnedRecipient`- value of reflection from tokens for recipient;

- `TransferToExcluded(address indexed sender, address indexed recipient, uint256 rOwnedSender, uint256 tOwnedRecipient, uint256 rOwnedRecipient);`

This event is emitted when user will transfer tokens to account that excluded from rewards.

Arguments:

-`sender`- address of account that transfer amount;

-`recipient`- address of account that get transfer's amount;

-`rOwnedSender`- value of reflection from tokens for sender;

-`tOwnedRecipient`- value of token from reflections for recipient;

-`rOwnedRecipient`- value of reflection from tokens for recipient;

- `TransferFromExcluded(address indexed sender, address indexed recipient, uint256 tOwnedSender, uint256 rOwnedSender, uint256 rOwnedRecipient);`

This event is emitted when user that excluded from rewards will transfer tokens.

Arguments:

-`sender`- address of account that transfer amount;

-`recipient`- address of account that get transfer's amount;

-`tOwnedSender`- value of token from reflections for sender;

-`rOwnedSender`- value of reflection from tokens for sender;

-`rOwnedRecipient`- value of reflection from tokens for recipient;

- `WithdrawLeftovers(address indexed recipient, uint256 amount);`

This event is emitted when owner withdraws leftovers from balance of the contract.

Arguments:

-`recipient`- address of account that get amount (address of current owner);

-`amount`- value of leftovers that withdraw from contract's balance;

- `WithdrawAlienToken(address indexed token, address indexed recipient, uint256 amount);`

This event is emitted when owner transfers alien tokens to other account from the contract.

Arguments:

-`token`- address of alien token (should not be zero's address or address of arcane token);

-`recipient`- address of account that get amount tokens;

-`amount`- value of alien tokens for transfer to other account;

- `AddLiquidity(uint256 amountToken, uint256 amountETH, uint256 liquidity);`

This event is emitted when call function addLiquidity after swap tokens and liquify.

Arguments:

-`amountToken`- amount of tokens after add liquidity;

-`amountETH`- amount of eth after add liquidity;

-`liquidity`- got value of liquidity from function `addLiquidityETH`;

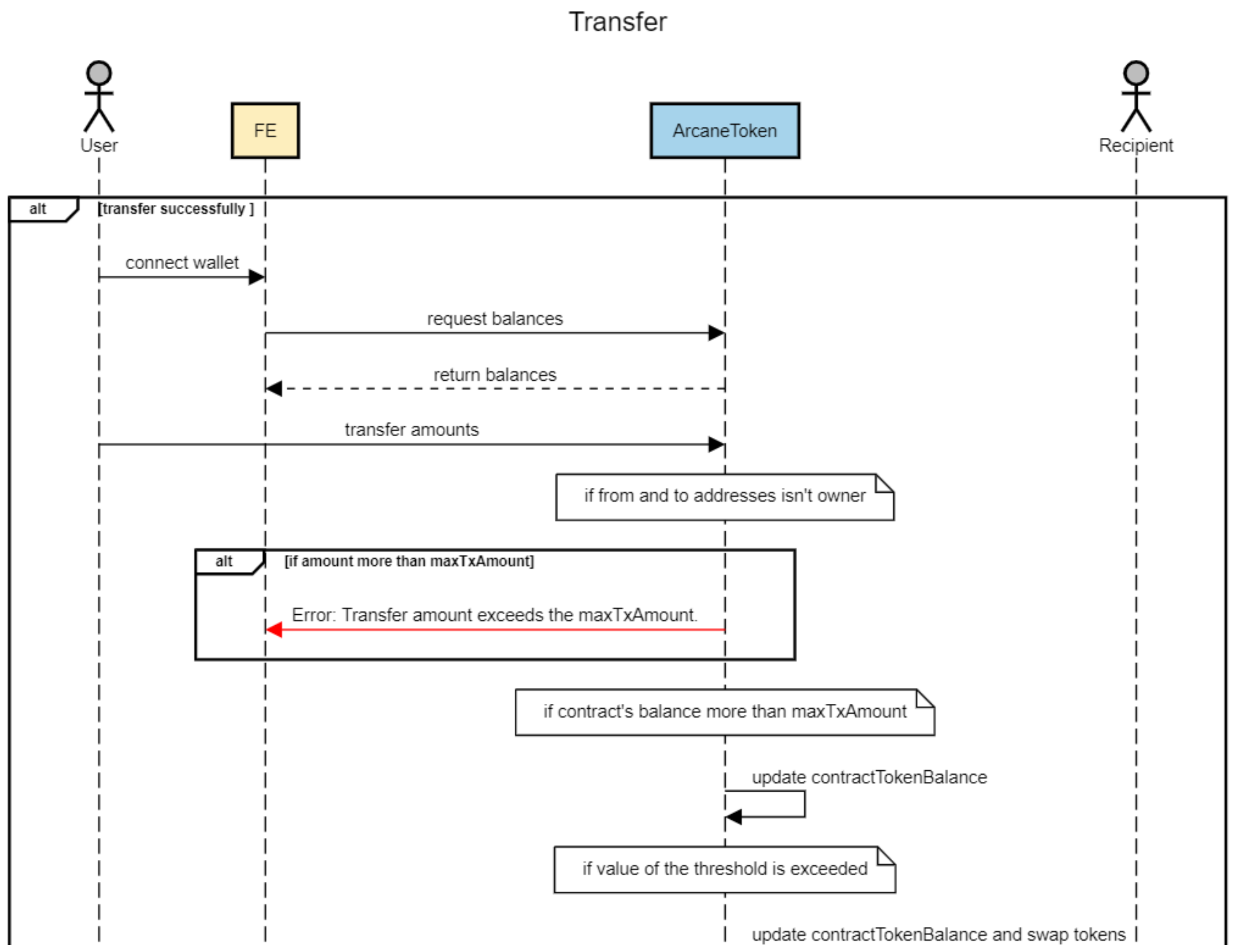
- `event ChangeRouter(address indexed router);`

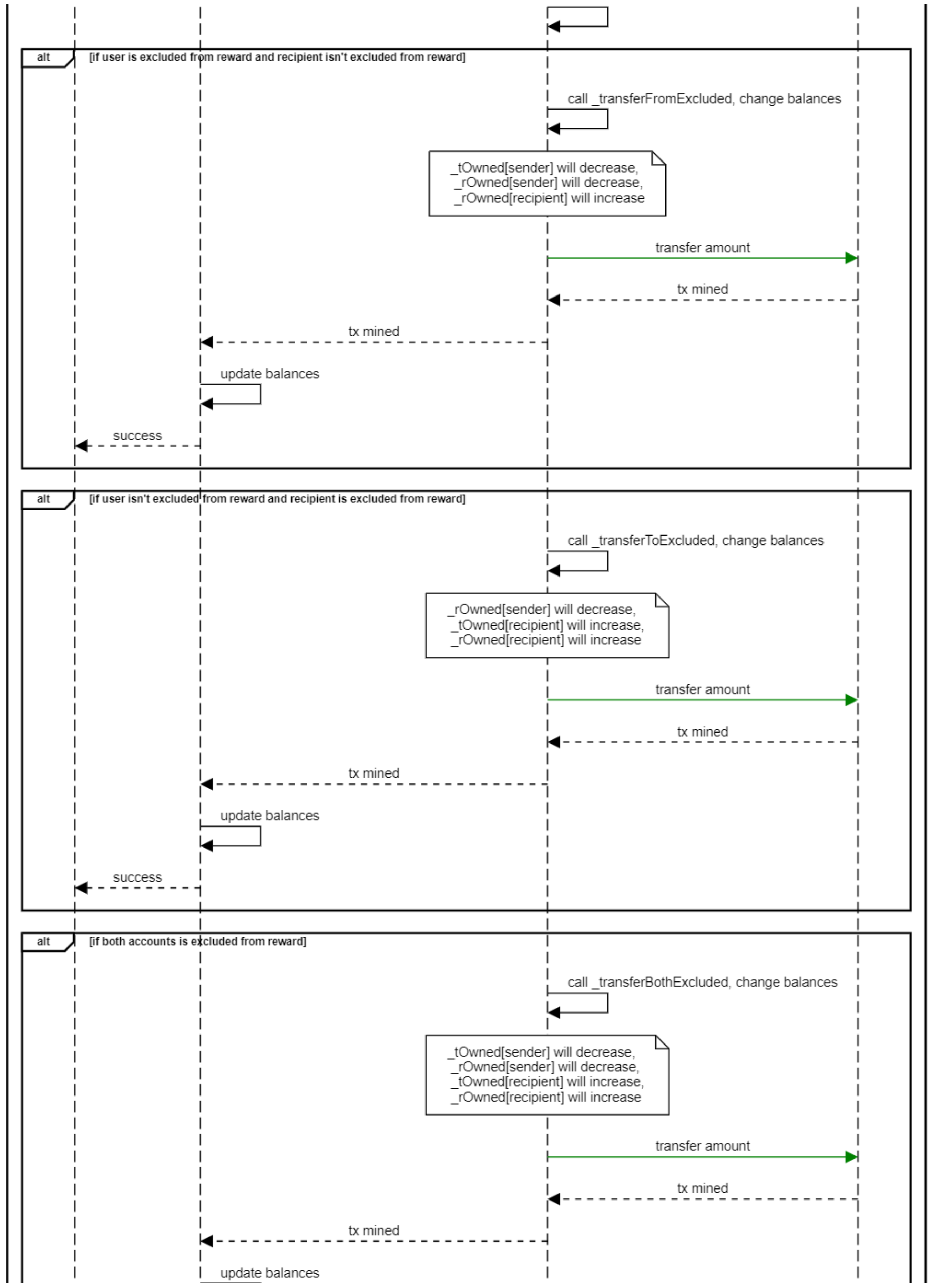
This event is emitted when address of router is changed

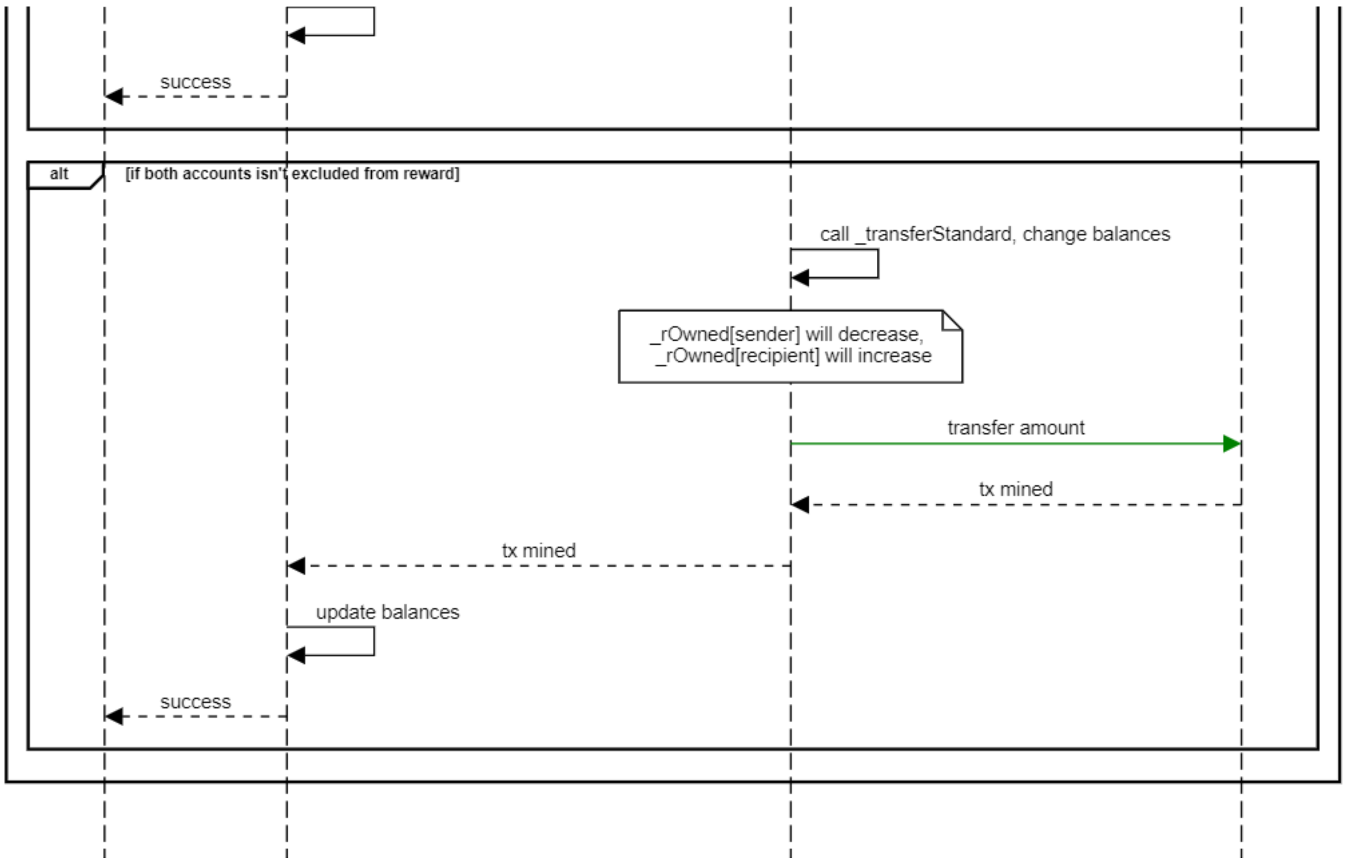
Arguments:

-`router` - address of `uniswapV2Router` contract

Sequence diagram


Transfer
User
FE
ArcaneToken
Recipient
alt
[transfer successfully]
connect wallet
request balances
return balances
transfer amounts
if from and to addresses isn't owner
alt
[if amount more than maxTxAmount]
Error: Transfer amount exceeds the maxTxAmount.
if contract's balance more than maxTxAmount
update contractTokenBalance
if value of the threshold is exceeded
update contractTokenBalance and swap tokens


alt
[if user is excluded from reward and recipient isn't excluded from reward]
call _transferFromExcluded, change balances
_tOwned[sender] will decrease,
_rOwned[sender] will decrease,
_rOwned[recipient] will increase
transfer amount
tx mined
tx mined
update balances
success
alt
[if user isn't excluded from reward and recipient is excluded from reward]
call _transferToExcluded, change balances
_rOwned[sender] will decrease,
_tOwned[recipient] will increase,
_rOwned[recipient] will increase
transfer amount
tx mined
tx mined
update balances
success
alt
[if both accounts is excluded from reward]
call _transferBothExcluded, change balances
_tOwned[sender] will decrease,
_rOwned[sender] will decrease,
_tOwned[recipient] will increase,
_rOwned[recipient] will increase
transfer amount
tx mined
tx mined
update balances


success
alt
[if both accounts isn't excluded from reward]
call _transferStandard, change balances
_rOwned[sender] will decrease,
_rOwned[recipient] will increase
transfer amount
tx mined
tx mined
update balances
success

Third party library

The smart-contract inherits next SC from the openzeppelin's libraries (v ^4.6.0):

- `OwnableUpgradeable.sol` - access control
- `ERC20Upgradeable.sol` - standard token ERC20
- `SafeMathUpgradeable.sol` - to safe work with math functions (prevent over/under flow)
- `IERC20.sol` - standard interface of ERC20

Upgradeable SCs

- ArcaneToken- using **TransparentUpgradeableProxy** pattern from package **@openzeppelin/truffle-upgrades**.

Risk and issues that can break the contract logic

- In the ArcaneToken contract in functions `includeInReward()` and `_getCurrentSupply()` can be out of gas because this functions have cycles with calculations (or a few actions).
- With the upgradeable of the contract, its size may exceeds the limit.
- The owner of contract ArcaneToken has many the permission without obtaining the consensus of the community.
- 3rd parties may be compromised that will lead to assets lost or stolen.
- The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function, in which the owner is the address for receiving tokens. As a result, a significant part of the tokens can accumulate at the address of the owner. If `_owner` is an external account, misusing its private key can have devastating consequences for the project as a whole.