Securosys SET token description

Blockchain Valley Ventures

October 24, 2018

Introduction

The SET security token is a non-voting share in Securosys AG. The SET security token can be exchanged to a Securosys share upon agreement with the company. When this happens the security tokens corresponding to the share are destroyed. To prove ownership of the tokens on a certain address, the token holder can be requested to send the tokens to be burned to the 0x0 address, from which the company than burns them using the burnFrom function. This Process might not be necessary if an off-chain token registry is held.
If an Securosys share is exchanged for SET security tokens, these new token need to be minted. The SET token also gives right to receive a dividend payment if the company decides so, the dividends can be payed out in either SET tokens, Ether (ETH) tokens or Fiat money.

Description

In order to fulfill the requirements of a security token the SET contract will inherit all functions from the here described token types, these functions allow for basic ERC-20 functionalities.They give SET functions to create the ownership structure, to mint/burn tokens as well as functions to pause the contract and pay dividends in Ether or native SET tokens (Fiat dividends are done off-chain).
The inheritance structure was chosen to allow for proven and tested token functions to be used in combination with the newly developed dividend paying system.

ERC-20

These are the standard functions of the ERC-20 token, they are used to keep track of balances, allow for transactions and for delegated transactions, the transfer functions are overwritten by the **pause** functions in order to allow for transaction freezing. There are 2 types of functions in solidity: the ones which change the state of the contract and the ones which only read the state, the read-only functions are called **view** and cost no gas to call. These functions can execute computations and return values. In this document all **view** functions will be labeled as such, all non labeled functions cost gas.

totalSupply, view

This functions returns the total supply of tokens currently in existence.

balanceOf, view

This function takes as parameter an address and returns the token balance of the given address.

transfer

This function takes as parameters the address of the receiver and the amount of tokens to be sent, the amount specified is sent from the address calling the function.

approve

This function takes as parameters the address which is given the right to spend the tokens and the amount of tokens that can be spend. By calling this function the address making the call can give another address the permission to spend tokens on its behalf with the *transferFrom* function. This function should only be called when allowance is 0, in other cases use increase/decrease Approval.

transferFrom

This function takes 3 parameters: the address from which to send the tokens, the address receiving the tokens and the amount of tokens to be transferred. The functions transfer the given amount from one address to the other. The amount to be transfers needs to be approved by the sender address through the *approve* function.

allowance, view

This function returns the amount of tokens which are allowed to be spent from one address by another address.

increaseApproval

This function has the same parameters as *approve* and can be called to increase the amount of tokens an owner allows to be spent by a spender.

decreaseApproval

This function has the same parameters as *approve* and can be called to decrease the amount of tokens an owner allows to be spent by a spender.

Ownable

This functions make the contract Ownable, meaning that there is one address with increased permissions. The owner, can mint/burn tokens as well as pause the contract and pay dividends. The owner is initially set to the address deploying the contract.
The "renounceOwnership" function has been taken out as it is not needed.
It is important to notice that the owner, due to his extended power, is a single point of failure of the entire contract, this requires extensive security measures when handling the owners private key.

onlyOwner

This is a modifier, these can be added to a function to be executed before the body of the function is reached. The onlyOwner modifiers requires the caller of a given function to be the owner of the contract.

transferOwnership

This function can be called by the owner to change the owner of the contract, the address given as parameter is set as the new owner and the current address calling the function is no longer the owner

Pausable

This functions allow the owner to pause the contract and lock tokens, if the contract is paused no more transfers are possible, this can be used if an upgrade of the contract is needed. In this case the owner would pause the contract, deploy a new contract and mint tokens on the new contract corresponding to the (frozen) balances of the old contract. The owner can also lock tokens of specific addresses until a given timestamp.
This functions overwrite the permission of the transfer and transferFrom function defined in the StandarToken contract.

pause

This function pauses the smart contract, when paused no transfers are possible. This function can only be called by the owner.

unpause

This function unpauses the smart contract. This function can only be called by the owner.

lockTokens

This function takes as input an array of addresses and an array of UNIX-Timestamps, each address has as its corresponding timeout date(address[i] gets timeout[i]) the value ad the same index. This function locks all transfers from the given addresses until the block with timestamp higher than the timeout value is minted. Note this function can only be called by the owner and lockups can NOT be overwritten even by the owner. This function can be called only once per address and is only intended for vesting-period lockups after the ICO.

Mintable

This functions allows the owner to create tokens, this functions are needed in the initial distribution of the token as well as when new tokens need to be created in exchange for company shares. If the owner decides that no more tokens need to be minted, he can block the minting, capping the total supply of tokens at the current total supply.

mint

This function takes two parameters: the address to receive the new tokens and the amount of tokens to be minted, this creates new tokens and adds them to balance of the given address, this increases the total supply. Only the owner can call this function.

finishMinting

This function can be called by the owner only and it blocks the minting function, after this function is called no more token will ever be minted.

Burnable

This function is used by the owner in two occasions: if an investor wants to change his tokens into company shares, in this case the tokens are burned upon issuance of the share. Or if a non-accredited investor has some tokens he is not supposed to have, in this case his tokens can be burned and new tokens can be minted to an allowed address.

burnFrom

This function takes as parameters the address from which tokens get burned and the amount of tokens to be burned. The function can only be called by the owner and it reduces the totalSupply of tokens. It is not required to give the owner allowance to burn the token. No account except the owner can burn tokens (not even his own).

Dividend

This functions can only be called by the owner and are used for on-chain dividend payments. The company has an address holding the reserve tokens minted after the Initial Distribution Phase. This address needs to be set in the contract before starting the dividend paying process, also the minimum threshold needs to be specified. Once the two parameters are set the owner depending on if the dividends are to be payed out in Ether or SET, deposits the Ether in the contract or allow SET to be moved out of the reserve by the owner. The owner than calls the *payDividend* functions once for each investor, this pays out the specified dividend. If too much Ether was deposited it can be withdrawn with the given function.

setEligibilityThreshold

This function can be called to set a threshold at which an address is eligible to receive dividend payments. This value applies universally to all token holders.

setTokenReserveAddress

This function is used to set the address of the token reserve. This is the address from which tokens will be moved to pay dividends.

approvePayoutFromReserve

This function needs to be called before the *payDividendFromReserve*. The function takes as parameter the total amount of tokens to be payed out in dividends in SET tokens. The function gives the owner of the contract the permission pay out the given amount in dividends from the reserve.

payDividendFromReserve

This function takes two parameters: the address of the receiver of the dividend payment and the amount to be payed out. This function transfers tokens from the reserve to the receiver.

depositEtherForDividends

This function is called to deposit Ether to the smart contract to be payed out in dividends, it takes as parameter the amount of ether to be deposited. When calling this function the correct amount of Ether needs to be sent in the Transaction.

payDividendInEther

This function takes two parameters: the address of the receiver of the dividend payment and the amount to be payed out. This function transfers the given amount of Ether from the balance of the contract to the receiver address.

withdrawEther

This function can be used to take Ether out of the contract in case to much Ether was deposited. It takes as parameter the amount of Ether to be taken out.

SET

This is the contract which will be deployed it inherits all functions from:

- StandartToken (ERC-20)
- OwnableToken
- PausableToken
- MintableToken
- BurnableToken
- DividendPayingToken

The SET token itself will only specify the name, the symbol and the decimals of the token.