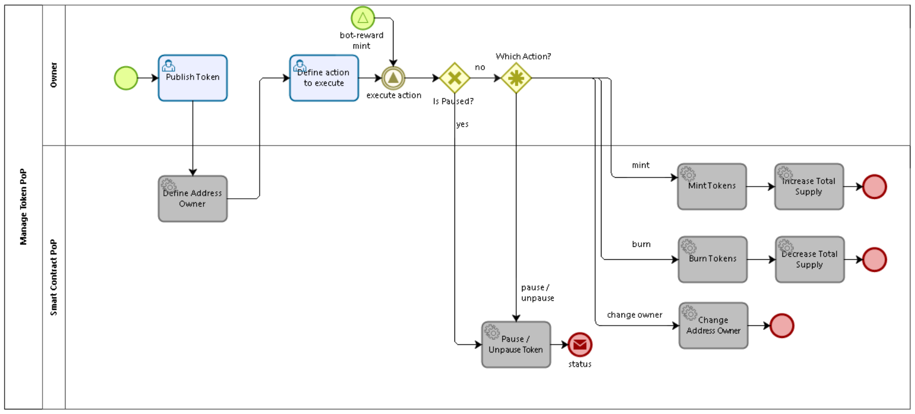
Must Currency

The **Must** currency is an **Must-20** template standard Ethereum, mintable and burnable, with owner access permissions and module pausable.
The **Must** was created to be a digital currency used within the permissioned blockchain. It offers mechanisms to handle most important **Must-20** transactions.
Within the permissioned blockchain ecosystem, only the bot-reward will execute mint **Must** currency .
The **Must** currency has two modules: Ownable and Pausable.
The Ownable module provides a basic access control mechanism, where there is an account (an owner) that can be granted exclusive access to specific functions. It's the only role set in **Must**.
The Pausable module allows to implement an emergency stop mechanism that can be triggered by an owner account.

Contract Address	0x948F964Bb4385d404f1a7F6Fe98bdD50c664643f

Manage Must Currency



Definitions

Name	It is still important to give it an identity.
Symbol	The symbol represents your brand.
Decimals	The divisibility will help us determine the lowest possible value of the Must. A divisibility of 0 will mean that the lowest value of the Must is 1. A divisibility of 2, on the other hand, means its lowest value will be 0.01. The maximum number of decimal places allowed is 18.

Total Supply	Number of Must Currency that will exist in the created ecosystem. This amount can be changed according to market needs, and can be increased or decreased (from the Mint and Burn functions respectively)

Functions

Internal functions from the Must-20 template standard:

Function	Description
`name() - string`	Returns the name of the coin.
`symbol() - string`	Returns the symbol of the coin.
`decimals() - uint8`	Returns the number of decimals used to get its user representation.
`totalSupply() - uint256`	Returns the amount of coins in existence.

Standard functions:

`constructor( string name, string symbol, uint8 decimals, uint256 totalSupply)`	The `constructor` function set the `name`, `symbol`, `decimals` and `totalSupply` of the Must.
`balanceOf(address _owner) - uint256`	Returns the amount of coins owned by an account (`_owner`).
`transfer(uint256 _value, address _to) - bool`	The method `transfer` is called by an account and transfers `_value` amount of coins to other address `_to`. Fire the `Transfer` event.
`transferFrom( address _from, address _to, uint256 _value )`	The method `transferFrom` allow one third account transfers `_value` amount of coins from other address `_from` to other address `_to`. Fire the `Transfer` event.
`approve(address _spender, uint256 _value) - bool`	The method `approve` allows other account `_spender` to withdraw from one account multiple times, up to the `_value` amount. Fire the`Approval` event.
`increaseApproval(address _spender, uint _addedValue)`	The method `increaseApproval` allows other account `_spender` to withdraw from one account multiple times, up to the `_addedValue` amount. Fire the`Approval` event with updated `value`
`decreaseApproval(address _spender, uint _subtractedValue)`	The method `decreaseApproval` reduces the value approved to `_spender` to withdraw from one account multiple times, substracting the `_subtractedValue` to the approval amount. If the `_subtractedValue` is bigger than previously approved the value will reduce to 0. . Fire the`Approval` event with updated `value`

`mintTo(uint256 _amount, address _to)`	The `mintTo` function creates `_amount`tokens and assigns them to account `_to`, increasing the total supply. Only owner can mint.
`burnFrom(uint256 _amount, address _from)`	The `burnFrom` function destroys `_amount` tokens from account `_from`, reducing the total supply. Only owner can burn.
`allowance(address _owner, address _spender) - uint256`	The view function `allowance` returns the amount which address `_spender` is still allowed to withdraw from `_owner`.
`transferFrom(address _from, address _to, uint256 _value) - bool`	Moves `_value` coins from address `_from`to address `_to` using the allowance mechanism. `_value` is then deducted from the caller's allowance.
`pause()`	Only owner can pause functions Must.
`unpause()`	Only owner can unpause functions Must.
`transferOwnership`()	The owner address can be changed with method `transferOwnership`.