OG ERC20

ALL 2.15

CODE, ABI, BYTECODE, RUNTIME

from vyper.interfaces import ERC20

implements: ERC20

event Transfer:

sender: indexed(address)

receiver: indexed(address)

value: uint256

event Approval:

owner: indexed(address)

spender: indexed(address)

value: uint256

name: public(String[64])

symbol: public(String[32])

decimals: public(uint256)

# NOTE: By declaring `balanceOf` as public, vyper automatically generates a 'balanceOf()' getter

# method to allow access to account balances.

# The \_KeyType will become a required parameter for the getter and it will return \_ValueType.

# See: https://vyper.readthedocs.io/en/v0.1.0-beta.8/types.html?highlight=getter#mappings

balanceOf: public(HashMap[address, uint256])

# By declaring `allowance` as public, vyper automatically generates the `allowance()` getter

allowance: public(HashMap[address, HashMap[address, uint256]])

# By declaring `totalSupply` as public, we automatically create the `totalSupply()` getter

totalSupply: public(uint256)

minter: address

@external

def \_\_init\_\_(\_name: String[64], \_symbol: String[32], \_decimals: uint256, \_supply: uint256):

init\_supply: uint256 = \_supply \* 10 \*\* \_decimals

self.name = \_name

self.symbol = \_symbol

self.decimals = \_decimals

self.balanceOf[msg.sender] = init\_supply

self.totalSupply = init\_supply

self.minter = msg.sender

log Transfer(ZERO\_ADDRESS, msg.sender, init\_supply)

@external

def transfer(\_to : address, \_value : uint256) -> bool:

"""

@dev Transfer token for a specified address

@param \_to The address to transfer to.

@param \_value The amount to be transferred.

"""

# NOTE: vyper does not allow underflows

# so the following subtraction would revert on insufficient balance

self.balanceOf[msg.sender] -= \_value

self.balanceOf[\_to] += \_value

log Transfer(msg.sender, \_to, \_value)

return True

@external

def transferFrom(\_from : address, \_to : address, \_value : uint256) -> bool:

"""

@dev Transfer tokens from one address to another.

@param \_from address The address which you want to send tokens from

@param \_to address The address which you want to transfer to

@param \_value uint256 the amount of tokens to be transferred

"""

# NOTE: vyper does not allow underflows

# so the following subtraction would revert on insufficient balance

self.balanceOf[\_from] -= \_value

self.balanceOf[\_to] += \_value

# NOTE: vyper does not allow underflows

# so the following subtraction would revert on insufficient allowance

self.allowance[\_from][msg.sender] -= \_value

log Transfer(\_from, \_to, \_value)

return True

@external

def approve(\_spender : address, \_value : uint256) -> bool:

"""

@dev Approve the passed address to spend the specified amount of tokens on behalf of msg.sender.

Beware that changing an allowance with this method brings the risk that someone may use both the old

and the new allowance by unfortunate transaction ordering. One possible solution to mitigate this

race condition is to first reduce the spender's allowance to 0 and set the desired value afterwards:

https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729

@param \_spender The address which will spend the funds.

@param \_value The amount of tokens to be spent.

"""

self.allowance[msg.sender][\_spender] = \_value

log Approval(msg.sender, \_spender, \_value)

return True

@external

def mint(\_to: address, \_value: uint256):

"""

@dev Mint an amount of the token and assigns it to an account.

This encapsulates the modification of balances such that the

proper events are emitted.

@param \_to The account that will receive the created tokens.

@param \_value The amount that will be created.

"""

assert msg.sender == self.minter

assert \_to != ZERO\_ADDRESS

self.totalSupply += \_value

self.balanceOf[\_to] += \_value

log Transfer(ZERO\_ADDRESS, \_to, \_value)

@internal

def \_burn(\_to: address, \_value: uint256):

"""

@dev Internal function that burns an amount of the token of a given

account.

@param \_to The account whose tokens will be burned.

@param \_value The amount that will be burned.

"""

assert \_to != ZERO\_ADDRESS

self.totalSupply -= \_value

self.balanceOf[\_to] -= \_value

log Transfer(\_to, ZERO\_ADDRESS, \_value)

@external

def burn(\_value: uint256):

"""

@dev Burn an amount of the token of msg.sender.

@param \_value The amount that will be burned.

"""

self.\_burn(msg.sender, \_value)

@external

def burnFrom(\_to: address, \_value: uint256):

"""

@dev Burn an amount of the token from a given account.

@param \_to The account whose tokens will be burned.

@param \_value The amount that will be burned.

"""

self.allowance[\_to][msg.sender] -= \_value

self.\_burn(\_to, \_value)

[{"name": "Transfer", "inputs": [{"name": "sender", "type": "address", "indexed": true}, {"name": "receiver", "type": "address", "indexed": true}, {"name": "value", "type": "uint256", "indexed": false}], "anonymous": false, "type": "event"}, {"name": "Approval", "inputs": [{"name": "owner", "type": "address", "indexed": true}, {"name": "spender", "type": "address", "indexed": true}, {"name": "value", "type": "uint256", "indexed": false}], "anonymous": false, "type": "event"}, {"stateMutability": "nonpayable", "type": "constructor", "inputs": [{"name": "\_name", "type": "string"}, {"name": "\_symbol", "type": "string"}, {"name": "\_decimals", "type": "uint256"}, {"name": "\_supply", "type": "uint256"}], "outputs": []}, {"stateMutability": "nonpayable", "type": "function", "name": "transfer", "inputs": [{"name": "\_to", "type": "address"}, {"name": "\_value", "type": "uint256"}], "outputs": [{"name": "", "type": "bool"}], "gas": 77310}, {"stateMutability": "nonpayable", "type": "function", "name": "transferFrom", "inputs": [{"name": "\_from", "type": "address"}, {"name": "\_to", "type": "address"}, {"name": "\_value", "type": "uint256"}], "outputs": [{"name": "", "type": "bool"}], "gas": 114965}, {"stateMutability": "nonpayable", "type": "function", "name": "approve", "inputs": [{"name": "\_spender", "type": "address"}, {"name": "\_value", "type": "uint256"}], "outputs": [{"name": "", "type": "bool"}], "gas": 37791}, {"stateMutability": "nonpayable", "type": "function", "name": "mint", "inputs": [{"name": "\_to", "type": "address"}, {"name": "\_value", "type": "uint256"}], "outputs": [], "gas": 79541}, {"stateMutability": "nonpayable", "type": "function", "name": "burn", "inputs": [{"name": "\_value", "type": "uint256"}], "outputs": [], "gas": 77898}, {"stateMutability": "nonpayable", "type": "function", "name": "burnFrom", "inputs": [{"name": "\_to", "type": "address"}, {"name": "\_value", "type": "uint256"}], "outputs": [], "gas": 115549}, {"stateMutability": "view", "type": "function", "name": "name", "inputs": [], "outputs": [{"name": "", "type": "string"}], "gas": 12798}, {"stateMutability": "view", "type": "function", "name": "symbol", "inputs": [], "outputs": [{"name": "", "type": "string"}], "gas": 10551}, {"stateMutability": "view", "type": "function", "name": "decimals", "inputs": [], "outputs": [{"name": "", "type": "uint256"}], "gas": 2628}, {"stateMutability": "view", "type": "function", "name": "balanceOf", "inputs": [{"name": "arg0", "type": "address"}], "outputs": [{"name": "", "type": "uint256"}], "gas": 2873}, {"stateMutability": "view", "type": "function", "name": "allowance", "inputs": [{"name": "arg0", "type": "address"}, {"name": "arg1", "type": "address"}], "outputs": [{"name": "", "type": "uint256"}], "gas": 3118}, {"stateMutability": "view", "type": "function", "name": "totalSupply", "inputs": [], "outputs": [{"name": "", "type": "uint256"}], "gas": 2718}]





Contract 2

from vyper.interfaces import ERC20

implements: ERC20

event Transfer:

sender: indexed(address)

receiver: indexed(address)

value: uint256

event Approval:

owner: indexed(address)

spender: indexed(address)

value: uint256

allowance: public(HashMap[address, HashMap[address, uint256]])

balanceOf: public(HashMap[address, uint256])

totalSupply: public(uint256)

nonces: public(HashMap[address, uint256])

DOMAIN\_SEPARATOR: public(bytes32)

DOMAIN\_TYPE\_HASH: constant(bytes32) = keccak256('EIP712Domain(string name,string version,uint256 chainId,address verifyingContract)')

PERMIT\_TYPE\_HASH: constant(bytes32) = keccak256("Permit(address owner,address spender,uint256 value,uint256 nonce,uint256 deadline)")

YFI: constant(address) = 0xdc7844Aaaf7a63F9b281ef0b35B3a4C9fd821c87

@external

def \_\_init\_\_():

self.DOMAIN\_SEPARATOR = keccak256(

concat(

DOMAIN\_TYPE\_HASH,

keccak256(convert("Woofy", Bytes[5])),

keccak256(convert("1", Bytes[1])),

convert(chain.id, bytes32),

convert(self, bytes32)

)

)

@view

@external

def name() -> String[5]:

return "Woofy"

@view

@external

def symbol() -> String[5]:

return "WOOFY"

@view

@external

def decimals() -> uint256:

return 12

@internal

def \_mint(receiver: address, amount: uint256):

assert not receiver in [self, ZERO\_ADDRESS]

self.balanceOf[receiver] += amount

self.totalSupply += amount

log Transfer(ZERO\_ADDRESS, receiver, amount)

@internal

def \_burn(sender: address, amount: uint256):

self.balanceOf[sender] -= amount

self.totalSupply -= amount

log Transfer(sender, ZERO\_ADDRESS, amount)

@internal

def \_transfer(sender: address, receiver: address, amount: uint256):

assert not receiver in [self, ZERO\_ADDRESS]

self.balanceOf[sender] -= amount

self.balanceOf[receiver] += amount

log Transfer(sender, receiver, amount)

@external

def transfer(receiver: address, amount: uint256) -> bool:

self.\_transfer(msg.sender, receiver, amount)

return True

@external

def transferFrom(sender: address, receiver: address, amount: uint256) -> bool:

self.allowance[sender][msg.sender] -= amount

self.\_transfer(sender, receiver, amount)

return True

@external

def approve(spender: address, amount: uint256) -> bool:

self.allowance[msg.sender][spender] = amount

log Approval(msg.sender, spender, amount)

return True

@external

def woof(amount: uint256 = MAX\_UINT256, receiver: address = msg.sender) -> bool:

mint\_amount: uint256 = min(amount, ERC20(YFI).balanceOf(msg.sender))

assert ERC20(YFI).transferFrom(msg.sender, self, mint\_amount)

self.\_mint(receiver, mint\_amount)

return True

@external

def unwoof(amount: uint256 = MAX\_UINT256, receiver: address = msg.sender) -> bool:

burn\_amount: uint256 = min(amount, self.balanceOf[msg.sender])

self.\_burn(msg.sender, burn\_amount)

assert ERC20(YFI).transfer(receiver, burn\_amount)

return True

@external

def permit(owner: address, spender: address, amount: uint256, expiry: uint256, signature: Bytes[65]) -> bool:

assert owner != ZERO\_ADDRESS # dev: invalid owner

assert expiry == 0 or expiry >= block.timestamp # dev: permit expired

nonce: uint256 = self.nonces[owner]

digest: bytes32 = keccak256(

concat(

b'\x19\x01',

self.DOMAIN\_SEPARATOR,

keccak256(

concat(

PERMIT\_TYPE\_HASH,

convert(owner, bytes32),

convert(spender, bytes32),

convert(amount, bytes32),

convert(nonce, bytes32),

convert(expiry, bytes32),

)

)

)

)

# NOTE: signature is packed as r, s, v

r: uint256 = convert(slice(signature, 0, 32), uint256)

s: uint256 = convert(slice(signature, 32, 32), uint256)

v: uint256 = convert(slice(signature, 64, 1), uint256)

assert ecrecover(digest, v, r, s) == owner # dev: invalid signature

self.allowance[owner][spender] = amount

self.nonces[owner] = nonce + 1

log Approval(owner, spender, amount)

return True

[{"name": "Transfer", "inputs": [{"name": "sender", "type": "address", "indexed": true}, {"name": "receiver", "type": "address", "indexed": true}, {"name": "value", "type": "uint256", "indexed": false}], "anonymous": false, "type": "event"}, {"name": "Approval", "inputs": [{"name": "owner", "type": "address", "indexed": true}, {"name": "spender", "type": "address", "indexed": true}, {"name": "value", "type": "uint256", "indexed": false}], "anonymous": false, "type": "event"}, {"stateMutability": "nonpayable", "type": "constructor", "inputs": [], "outputs": []}, {"stateMutability": "view", "type": "function", "name": "name", "inputs": [], "outputs": [{"name": "", "type": "string"}], "gas": 5916}, {"stateMutability": "view", "type": "function", "name": "symbol", "inputs": [], "outputs": [{"name": "", "type": "string"}], "gas": 5946}, {"stateMutability": "view", "type": "function", "name": "decimals", "inputs": [], "outputs": [{"name": "", "type": "uint256"}], "gas": 348}, {"stateMutability": "nonpayable", "type": "function", "name": "transfer", "inputs": [{"name": "receiver", "type": "address"}, {"name": "amount", "type": "uint256"}], "outputs": [{"name": "", "type": "bool"}], "gas": 78588}, {"stateMutability": "nonpayable", "type": "function", "name": "transferFrom", "inputs": [{"name": "sender", "type": "address"}, {"name": "receiver", "type": "address"}, {"name": "amount", "type": "uint256"}], "outputs": [{"name": "", "type": "bool"}], "gas": 116239}, {"stateMutability": "nonpayable", "type": "function", "name": "approve", "inputs": [{"name": "spender", "type": "address"}, {"name": "amount", "type": "uint256"}], "outputs": [{"name": "", "type": "bool"}], "gas": 37881}, {"stateMutability": "nonpayable", "type": "function", "name": "woof", "inputs": [], "outputs": [{"name": "", "type": "bool"}]}, {"stateMutability": "nonpayable", "type": "function", "name": "woof", "inputs": [{"name": "amount", "type": "uint256"}], "outputs": [{"name": "", "type": "bool"}]}, {"stateMutability": "nonpayable", "type": "function", "name": "woof", "inputs": [{"name": "amount", "type": "uint256"}, {"name": "receiver", "type": "address"}], "outputs": [{"name": "", "type": "bool"}]}, {"stateMutability": "nonpayable", "type": "function", "name": "unwoof", "inputs": [], "outputs": [{"name": "", "type": "bool"}]}, {"stateMutability": "nonpayable", "type": "function", "name": "unwoof", "inputs": [{"name": "amount", "type": "uint256"}], "outputs": [{"name": "", "type": "bool"}]}, {"stateMutability": "nonpayable", "type": "function", "name": "unwoof", "inputs": [{"name": "amount", "type": "uint256"}, {"name": "receiver", "type": "address"}], "outputs": [{"name": "", "type": "bool"}]}, {"stateMutability": "nonpayable", "type": "function", "name": "permit", "inputs": [{"name": "owner", "type": "address"}, {"name": "spender", "type": "address"}, {"name": "amount", "type": "uint256"}, {"name": "expiry", "type": "uint256"}, {"name": "signature", "type": "bytes"}], "outputs": [{"name": "", "type": "bool"}], "gas": 91134}, {"stateMutability": "view", "type": "function", "name": "allowance", "inputs": [{"name": "arg0", "type": "address"}, {"name": "arg1", "type": "address"}], "outputs": [{"name": "", "type": "uint256"}], "gas": 3088}, {"stateMutability": "view", "type": "function", "name": "balanceOf", "inputs": [{"name": "arg0", "type": "address"}], "outputs": [{"name": "", "type": "uint256"}], "gas": 2903}, {"stateMutability": "view", "type": "function", "name": "totalSupply", "inputs": [], "outputs": [{"name": "", "type": "uint256"}], "gas": 2718}, {"stateMutability": "view", "type": "function", "name": "nonces", "inputs": [{"name": "arg0", "type": "address"}], "outputs": [{"name": "", "type": "uint256"}], "gas": 2963}, {"stateMutability": "view", "type": "function", "name": "DOMAIN\_SEPARATOR", "inputs": [], "outputs": [{"name": "", "type": "bytes32"}], "gas": 2778}]



