Wrapped XMG (WXMG)

About WXMG:

WXMG is a decentralized token on Binance Smart Chain. It's a wrapped version of Magi that can be wrapped using Magi Bridge. You can trade WXMG and stake WXMG-BNB LP tokens and earn WXMG with 100% APR.

Tokenization:

Contract Address: 0xeC159cd31964d7E64225F52757d0055f0beEA5c8
Name: Wrapped XMG
Symbol: WXMG
Total Supply: 25,000,000 WXMG
Maximal Supply: 25,000,000 WXMG
Decimals: 8
Minting: None
Burning: None
Transaction Time: 3 seconds
Transaction Fee: 0.0001 BNB at lowest

Contacts:

E-mail: info@magibridge.com - Odpovídáme do 24 hodin.
Discord: https://discord.com/invite/yvdz3M4HbG
Instagram: https://instagram.com/magicoinxmg

You can support this project here.

Contract Source Code:

WXMG
Public: balances: mapping(address=>uint) allowance: mapping(address=>mapping(address=>uint)) totalSupply: uint name: string symbol: string decimals: uint owner: address
Public: <<event>> Transfer(from: address, to: address, value: uint) <<event>> Approval(owner: address, spender: address, value: uint) <<event>> OwnershipTransferred(previousOwner: address, newOwner: address) <<modifier>> onlyOwner() constructor() balanceOf(owner: address): uint transfer(to: address, value: uint): bool transferFrom(from: address, to: address, value: uint): bool approve(spender: address, value: uint): bool Ownable() transferOwnership(newOwner: address)

```
/**
 *Submitted for verification at BscScan.com on 2022-06-06
*/

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.14;

contract WXMG {
    mapping(address => uint) public balances;
    mapping(address => mapping(address => uint)) public allowance;
    uint public totalSupply = 25000000 * 10 ** 8;
    string public name = "Wrapped XMG";
    string public symbol = "WXMG";
    uint public decimals = 8;

    event Transfer(address indexed from, address indexed to, uint value);
    event Approval(address indexed owner, address indexed spender, uint value);

    constructor() {
        balances[msg.sender] = totalSupply;
    }

    function balanceOf(address owner) public returns(uint) {
        return balances[owner];
    }

    function transfer(address to, uint value) public returns(bool) {
        require(balanceOf(msg.sender) >= value, 'balance too low');
        balances[to] += value;
        balances[msg.sender] -= value;
       emit Transfer(msg.sender, to, value);
        return true;
    }

    function transferFrom(address from, address to, uint value) public returns(bool) {
        require(balanceOf(from) >= value, 'balance too low');
        require(allowance[from][msg.sender] >= value, 'allowance too low');
        balances[to] += value;
        balances[from] -= value;
        emit Transfer(from, to, value);
        return true;
    }

    function approve(address spender, uint value) public returns (bool) {
        allowance[msg.sender][spender] = value;
        emit Approval(msg.sender, spender, value);
        return true;
    }

  address public owner;

  event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

  function Ownable() public {
    owner = msg.sender;
  }

  modifier onlyOwner() {
    require(msg.sender == owner);
     _;
  }

  function transferOwnership(address newOwner) public onlyOwner {
    require(newOwner != address(0));
    emit OwnershipTransferred(owner, newOwner);
    owner = newOwner;
  }
}
```