成都链安
BEOSIN

Smart contract security audit report

Audit Number：202008281040

Smart Contract Name：

Darwinia Commitment Token (KTON)

Smart Contract Address：

0x9F284E1337A815fe77D2Ff4aE46544645B20c5ff

Smart Contract Address Link：

https://etherscan.io/address/0x9f284e1337a815fe77d2ff4ae46544645b20c5ff#code

Start Date：2020.08.26

Completion Date：2020.08.28

Overall Result：Pass（Merit）

Audit Team: Beosin (Chengdu LianAn) Technology Co. Ltd.

Audit Categories and Results:

No.	Categories	Subitems	Results
1	Coding Conventions	ERC20 Token Standards	Pass
		Compiler Version Security	Pass
		Visibility Specifiers	Pass
		Gas Consumption	Pass
		SafeMath Features	Pass
		Fallback Usage	Pass
		tx.origin Usage	Pass
		Deprecated Items	Pass
		Redundant Code	Pass
		Overriding Variables	Pass
2	Function Call Audit	Authorization of Function Call	Pass
		Low-level Function (call/delegatecall) Security	Pass
		Returned Value Security	Pass
		selfdestruct Function Security	Pass

3	Business Security	Access Control of Owner	Pass
		Business Logics	Pass
		Business Implementations	Pass
4	Integer Overflow/Underflow	-	Pass
5	Reentrancy	-	Pass
6	Exceptional Reachable State	-	Pass
7	Transaction-Ordering Dependence	-	Pass
8	Block Properties Dependence	-	Pass
9	Pseudo-random Number Generator (PRNG)	-	Pass
10	DoS (Denial of Service)	-	Pass
11	Token Vesting Implementation	-	**Missing**
12	Fake Deposit	-	Pass
13	event security	-	Pass

Note: Audit results and suggestions in code comments

Disclaimer: This audit is only applied to the type of auditing specified in this report and the scope of given in the results table. Other unknown security vulnerabilities are beyond auditing responsibility. Beosin (Chengdu LianAn) Technology only issues this report based on the attacks or vulnerabilities that already existed or occurred before the issuance of this report. For the emergence of new attacks or vulnerabilities that exist or occur in the future, Beosin (Chengdu LianAn) Technology lacks the capability to judge its possible impact on the security status of smart contracts, thus taking no responsibility for them. The security audit analysis and other contents of this report are based solely on the documents and materials that the contract provider has provided to Beosin (Chengdu LianAn) Technology before the issuance of this report, and the contract provider warrants that there are no missing, tampered, deleted; if the documents and materials provided by the contract provider are missing, tampered, deleted, concealed or reflected in a situation that is inconsistent with the actual situation, or if the documents and materials provided are changed after the issuance of this report, Beosin (Chengdu LianAn) Technology assumes no responsibility for the resulting loss or adverse effects. The audit report issued by Beosin (Chengdu LianAn) Technology is based on the documents and materials provided by the contract provider, and relies on the technology currently possessed by Beosin (Chengdu LianAn). Due to the technical limitations of any organization, this report conducted by Beosin (Chengdu LianAn) still has the possibility that the entire risk cannot be completely detected. Beosin (Chengdu LianAn) disclaims any liability for the resulting losses.

The final interpretation of this statement belongs to Beosin (Chengdu LianAn).

Audit Results Explained:

Beosin (Chengdu LianAn) Technology has used several methods including Formal Verification, Static Analysis, Typical Case Testing and Manual Review to audit three major aspects of smart contract KTON, including Coding Standards, Security, and Business Logic. **KTON contract passed all audit items. The overall result is Pass (Merit). The smart contract is able to function properly.** Please find below the basic information of the smart contract:

1、Basic Token Information

Token name	Darwinia Commitment Token
Token symbol	KTON
decimals	18
totalSupply	About 53 Thousand(Destroyable, Mintable without limitation)
Token type	ERC20

Table 1 – Basic Token Information

2、Token Vesting Information

Missing

3、Other Audit Suggestions

- Variable type

  As shown in Figure 1 and 2 below, the contract uses bytes32 type when defining token symbol, uses uint256 when defining token decimals, and uses bytes32 when defining token name, which does not conform to ERC20 standard.

```
238 contract DSToken is DSTokenBase(0), DSStop {
239
240     bytes32  public  symbol;
241     uint256  public  decimals = 18; // standard token precision. override to customize
242
```

Figure 1 symbol, decimals variables source code

```
306     // Optional token name
307     bytes32   public  name = "";
308
309     function setName(bytes32 name_) public auth {
310         name = name_;
311     }
312 }
313
```

Figure 2 name variable source code

Modify recommendation： It is recommended to define variable types according to ERC20 standards.

- mint function

  As shown in Figure 4 and 5 below, the mint function is used to mint tokens, the authorized caller can call this function to mint tokens of the specified address and the tokens can be minted without limit.

```
291     function mint(address guy, uint wad) public auth stoppable {
292         _balances[guy] = add(_balances[guy], wad);
293         _supply = add(_supply, wad);
294         emit Mint(guy, wad);
295     }
```

Figure 4 mint function source code

Modify recommendation： Setting the maximum token total supply limit to a hard cap is recommended.

- Compiler warning

  The minimum version of the compiler declared in the contract is 0.4.23, and the higher 0.4.24 version of the compiler is used when deploying the contract on the main network. The compiler warnings shown in Figure 4 below will occur when compiling the contract with the 0.4.24 version of the compiler.

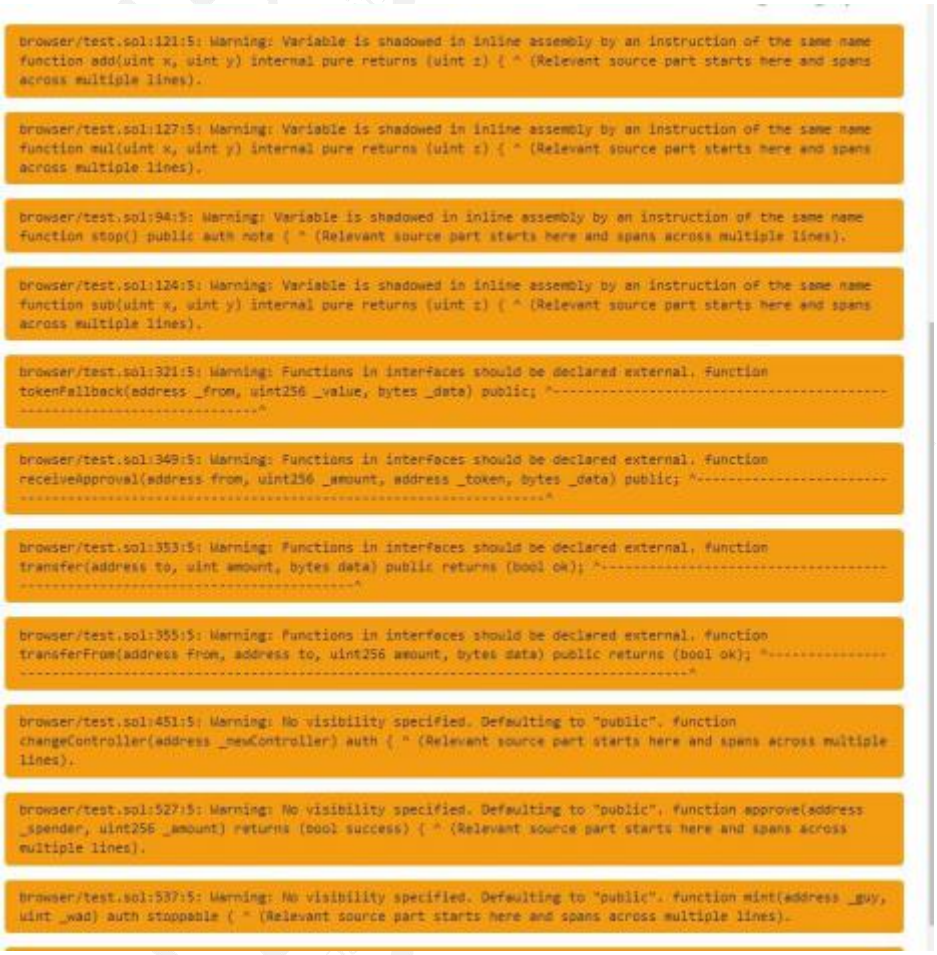

Figure 4 compiler warnings source code

Modify recommendation： It is recommended to improve the code to eliminate compiler warnings.

Audited Source Code with Comments:

```
/**
 *Submitted for verification at Etherscan.io on 2018-09-26
*/

pragma solidity ^0.4.23; // Beosin (Chengdu LianAn) // Fixing compiler version is recommended.
```

```
contract DSAuthority {
    // Beosin (Chengdu LianAn) // Define interface function.
    function canCall(
        address src, address dst, bytes4 sig
    ) public view returns (bool);
}

contract DSAuthEvents {
    event LogSetAuthority (address indexed authority); // Beosin (Chengdu LianAn) // Declare the event 'LogSetAuthority'.
    event LogSetOwner     (address indexed owner); // Beosin (Chengdu LianAn) // Declare the event 'LogSetOwner'.
}

contract DSAuth is DSAuthEvents {
    DSAuthority  public  authority;
    address      public  owner; // Beosin (Chengdu LianAn) // Declare the variable 'owner' for storing the address of contract owner.
    // Beosin (Chengdu LianAn) // Constructor, initialize the contract owner.
    constructor() public {
        owner = msg.sender;
        emit LogSetOwner(msg.sender); // Beosin (Chengdu LianAn) // Trigger the event 'LogSetOwner'.
    }

    function setOwner(address owner_)
        public
        auth
    {
        owner = owner_; // Beosin (Chengdu LianAn) // The 'owner' is set to 'owner_'.
        emit LogSetOwner(owner); // Beosin (Chengdu LianAn) // Trigger the event 'LogSetOwner'.
    }

    function setAuthority(DSAuthority authority_)
        public
        auth
    {
        authority = authority_; // Beosin (Chengdu LianAn) // The 'authority' is set to 'authority_'.
        emit LogSetAuthority(authority); // Beosin (Chengdu LianAn) // Trigger the event 'LogSetAuthority'.
    }
    // Beosin (Chengdu LianAn) // Modifier, require that the function caller should passed the check of 'isAuthorized'.
    modifier auth {
        require(isAuthorized(msg.sender, msg.sig));
        _;
    }
    // Beosin (Chengdu LianAn) // Check whether the specified address 'src' is authorized.
    function isAuthorized(address src, bytes4 sig) internal view returns (bool) {
        if (src == address(this)) {
```

```
                return true; // Beosin (Chengdu LianAn) // If 'src' is this contract address, return true.
            } else if (src == owner) {
                return true; // Beosin (Chengdu LianAn) // If 'src' is contract owner, return true.
            } else if (authority == DSAuthority(0)) {
                return false; // Beosin (Chengdu LianAn) // If 'authority' is zero address, return false.
            } else {
                return authority.canCall(src, this, sig); // Beosin (Chengdu LianAn) // Otherwise, return the result of calling the 'canCall' function of the authority contract.
            }
        }
}

contract DSNote {
    event LogNote(
        bytes4   indexed  sig,
        address  indexed  guy,
        bytes32  indexed  foo,
        bytes32  indexed  bar,
        uint              wad,
        bytes             fax
    ) anonymous; // Beosin (Chengdu LianAn) // Declare the events 'LogNote'.
    // Beosin (Chengdu LianAn) // Modifier, record the call information of the modified function.
    modifier note {
        bytes32 foo;
        bytes32 bar;

        assembly {
            foo := calldataload(4)
            bar := calldataload(36)
        }

        emit LogNote(msg.sig, msg.sender, foo, bar, msg.value, msg.data); // Beosin (Chengdu LianAn) // Trigger the event 'LogNote'.

        _;
    }
}

contract DSStop is DSNote, DSAuth {

    bool public stopped; // Beosin (Chengdu LianAn) // Declare the variable 'stopped' for storing the contract stopped status
    // Beosin (Chengdu LianAn) // Modifier, make a function callable only when the contract is not stopped.
    modifier stoppable {
        require(!stopped);
        _;
    }
    // Beosin (Chengdu LianAn) // Change the variable 'stopped' to true.
```

```
    function stop() public auth note {
        stopped = true;
    }
    // Beosin (Chengdu LianAn) // Change the variable 'stopped' to false.
    function start() public auth note {
        stopped = false;
    }

}

contract ERC20Events {
    event Approval(address indexed src, address indexed guy, uint wad); // Beosin (Chengdu LianAn) // Declare the event 'Approval'.
    event Transfer(address indexed src, address indexed dst, uint wad); // Beosin (Chengdu LianAn) // Declare the event 'Transfer'.
}

contract ERC20 is ERC20Events {
    // Beosin (Chengdu LianAn) // Define the function interfaces required in the ERC20 Token Standard.
    function totalSupply() public view returns (uint);
    function balanceOf(address guy) public view returns (uint);
    function allowance(address src, address guy) public view returns (uint);

    function approve(address guy, uint wad) public returns (bool);
    function transfer(address dst, uint wad) public returns (bool);
    function transferFrom(
        address src, address dst, uint wad
    ) public returns (bool);
}

contract DSMath {
    function add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x);
    }
    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x);
    }
    function mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x);
    }

    function min(uint x, uint y) internal pure returns (uint z) {
        return x <= y ? x : y;
    }
    function max(uint x, uint y) internal pure returns (uint z) {
        return x >= y ? x : y;
    }
    function imin(int x, int y) internal pure returns (int z) {
```

```
        return x <= y ? x : y;
    }
    function imax(int x, int y) internal pure returns (int z) {
        return x >= y ? x : y;
    }

    uint constant WAD = 10 ** 18;
    uint constant RAY = 10 ** 27;

    function wmul(uint x, uint y) internal pure returns (uint z) {
        z = add(mul(x, y), WAD / 2) / WAD;
    }
    function rmul(uint x, uint y) internal pure returns (uint z) {
        z = add(mul(x, y), RAY / 2) / RAY;
    }
    function wdiv(uint x, uint y) internal pure returns (uint z) {
        z = add(mul(x, WAD), y / 2) / y;
    }
    function rdiv(uint x, uint y) internal pure returns (uint z) {
        z = add(mul(x, RAY), y / 2) / y;
    }

    // This famous algorithm is called "exponentiation by squaring"
    // and calculates x^n with x as fixed-point and n as regular unsigned.
    //
    // It's O(log n), instead of O(n) for naive repeated multiplication.
    //
    // These facts are why it works:
    //
    //  If n is even, then x^n = (x^2)^(n/2).
    //  If n is odd,  then x^n = x * x^(n-1),
    //   and applying the equation for even x gives
    //    x^n = x * (x^2)^((n-1) / 2).
    //
    //  Also, EVM division is flooring and
    //    floor[(n-1) / 2] = floor[n / 2].
    //
    function rpow(uint x, uint n) internal pure returns (uint z) {
        z = n % 2 != 0 ? x : RAY;

        for (n /= 2; n != 0; n /= 2) {
            x = rmul(x, x);

            if (n % 2 != 0) {
                z = rmul(z, x);
            }
        }
    }
```

```
}


contract DSTokenBase is ERC20, DSMath {
    uint256                                            supply; // Beosin (Chengdu LianAn) //
Declare the variable '_supply' for storing the total supply of token.
    mapping (address => uint256)                       balances; // Beosin (Chengdu LianAn) // Declare
the variable '_balances' for storing token balance of corresponding address.
    mapping (address => mapping (address => uint256))  approvals; // Beosin (Chengdu LianAn) // Declare
the mapping variable '_approvals' for storing the allowance between two addresses.
    // Beosin (Chengdu LianAn) // Constructor, initialize the token total supply and send all tokens to
deployer.
    constructor(uint supply) public {
        balances[msg.sender] = supply;
        supply = supply;
    }

    function totalSupply() public view returns (uint) {
        return  supply;
    }
    function balanceOf(address src) public view returns (uint) {
        return _balances[src];
    }
    function allowance(address src, address guy) public view returns (uint) {
        return _approvals[src][guy];
    }

    function transfer(address dst, uint wad) public returns (bool) {
        return transferFrom(msg.sender, dst, wad);
    }

    function transferFrom(address src, address dst, uint wad)
        public
        returns (bool)
    {
        if (src != msg.sender) {
            _approvals[src][msg.sender] = sub(_approvals[src][msg.sender], wad); // Beosin (Chengdu
LianAn) // If the specified address 'src' is not the caller address, alter the allowance which 'src' allowed to
caller.
        }

        _balances[src] = sub(_balances[src], wad); // Beosin (Chengdu LianAn) // Alter the token balance of
'src'.
        _balances[dst] = add(_balances[dst], wad); // Beosin (Chengdu LianAn) // Alter the token balance of
'dst'.

        emit Transfer(src, dst, wad); // Beosin (Chengdu LianAn) // Trigger the event 'Transfer'.
```

```
        return true;
    }
    // Beosin (Chengdu LianAn) // The 'approve' function, 'msg.sender' allows the specified amount of tokens to a specified address.
    // Beosin (Chengdu LianAn) // It is recommended that users reset the allowance to zero, and then set a new allowance.
    function approve(address guy, uint wad) public returns (bool) {
        approvals[msg.sender][guy] = wad; // Beosin (Chengdu LianAn) // The allowance which caller allowed to 'guy' is set to 'wad'.

        emit Approval(msg.sender, guy, wad); // Beosin (Chengdu LianAn) // Trigger the event 'Approval'.

        return true;
    }
}

contract DSToken is DSTokenBase(0), DSStop {

    bytes32    public    symbol; // Beosin (Chengdu LianAn) // Declare the variable 'symbol' for storing the token symbol. Note: it is recommended to use string type.
    uint256    public    decimals = 18; // standard token precision. override to customize // Beosin (Chengdu LianAn) // Declare the variable 'decimals' for storing the token decimals, there are 18 decimals as default. Note: it is recommended to use uint8 type.
    // Beosin (Chengdu LianAn) // Constructor, initialize the token symbol.
    constructor(bytes32 symbol_) public {
        symbol = symbol_;
    }

    event Mint(address indexed guy, uint wad); // Beosin (Chengdu LianAn) // Declare the event 'Mint'.
    event Burn(address indexed guy, uint wad); // Beosin (Chengdu LianAn) // Declare the event 'Burn'.

    function approve(address guy) public stoppable returns (bool) {
        return super.approve(guy, uint(-1)); // Beosin (Chengdu LianAn) // Execute the function 'approve' of the parent contract to token approve, allowance 2**256-1, return the result.
    }
    // Beosin (Chengdu LianAn) // Rewrite the function 'approve', added the modifier 'stoppable' for the authorized caller controlling the approve function.
    function approve(address guy, uint wad) public stoppable returns (bool) {
        return super.approve(guy, wad); // Beosin (Chengdu LianAn) // Execute the function 'approve' of the parent contract to token approve, return the result.
    }
    // Beosin (Chengdu LianAn) // Rewrite the function 'transferFrom', added the modifier 'stoppable' for the authorized caller controlling the transferFrom function.
    function transferFrom(address src, address dst, uint wad)
        public
        stoppable
        returns (bool)
    {
```

```
        if (src != msg.sender &&  approvals[src][msg.sender] != uint(-1)) { // Beosin (Chengdu LianAn) // If transfer source address 'src' is not the caller and the allowance which 'src' allowed to caller is not the same as 2**256 -1, update the allowance which 'src' allowed to caller.
            approvals[src][msg.sender] = sub(  approvals[src][msg.sender], wad);
        }

        balances[src] = sub(  balances[src], wad); // Beosin (Chengdu LianAn) // Alter the token balance of 'src'.
        balances[dst] = add(  balances[dst], wad); // Beosin (Chengdu LianAn) // Alter the token balance of 'dst'.

        emit Transfer(src, dst, wad); // Beosin (Chengdu LianAn) // Trigger the event 'Transfer'.

        return true;
    }
    // Beosin (Chengdu LianAn) // The 'push' function, the caller transfers the specified amount of tokens to the specified address.
    function push(address dst, uint wad) public {
        transferFrom(msg.sender, dst, wad);
    }
    // Beosin (Chengdu LianAn) // The 'pull' function, the caller as a delegate of 'src' transfers the specified amount of tokens to itself.
    function pull(address src, uint wad) public {
        transferFrom(src, msg.sender, wad);
    }
    // Beosin (Chengdu LianAn) // The 'move' function, the caller as a delegate of 'src' transfers the specified amount of tokens to the specified address.
    function move(address src, address dst, uint wad) public {
        transferFrom(src, dst, wad);
    }
    // Beosin (Chengdu LianAn) // The 'mint' function, the caller mint the specified amount of tokens to itself.
    function mint(uint wad) public {
        mint(msg.sender, wad);
    }
    // Beosin (Chengdu LianAn) // The 'burn' function, the caller destroy the specified amount of tokens of itself.
    function burn(uint wad) public {
        burn(msg.sender, wad);
    }
    // Beosin (Chengdu LianAn) // The 'mint' function, the caller mint the specified amount of tokens to the specified address.
    function mint(address guy, uint wad) public auth stoppable {
        _balances[guy] = add(_balances[guy], wad); // Beosin (Chengdu LianAn) // Alter the token balance of 'guy'.
        _supply = add(_supply, wad); // Beosin (Chengdu LianAn) // Alter the token total supply.
        emit Mint(guy, wad); // Beosin (Chengdu LianAn) // Trigger the event 'Mint'.
    }
```

```
    // Beosin (Chengdu LianAn) // The 'burn' function, the caller as a delegate of 'guy' destroy the specified amount of tokens of the specified address.
    function burn(address guy, uint wad) public auth stoppable {
        if (guy != msg.sender &&   approvals[guy][msg.sender] != uint(-1)) { // Beosin (Chengdu LianAn) // If destroy source address 'guy' is not the caller and the allowance which 'guy' allowed to caller is not the same as 2**256 -1, update the allowance which 'guy' allowed to caller.
            approvals[guy][msg.sender] = sub(  approvals[guy][msg.sender], wad);
        }

        balances[guy] = sub(  balances[guy], wad); // Beosin (Chengdu LianAn) // Alter the token balance of 'guy'.
        supply = sub(  supply, wad); // Beosin (Chengdu LianAn) // Alter the token total supply.
        emit Burn(guy, wad); // Beosin (Chengdu LianAn) // Trigger the event 'Burn'.
    }

    // Optional token name
    bytes32    public    name = "";

    function setName(bytes32 name  ) public auth {
        name = name_; // Beosin (Chengdu LianAn) // Set token name to 'name_'.
    }
}

 /*
 * Contract that is working with ERC223 tokens
 * https://github.com/ethereum/EIPs/issues/223
 */
// Beosin (Chengdu LianAn) // Define interface function.
/// @title ERC223ReceivingContract - Standard contract implementation for compatibility with ERC223 tokens.
contract ERC223ReceivingContract {

    /// @dev Function that is called when a user or another contract wants to transfer funds.
    /// @param _from Transaction initiator, analogue of msg.sender
    /// @param _value Number of tokens to transfer.
    /// @param _data Data containig a function signature and/or parameters
    function tokenFallback(address _from, uint256 _value, bytes _data) public;

}
// Beosin (Chengdu LianAn) // Define interface function.
/// @dev The token controller contract must implement these functions
contract TokenController {
    /// @notice Called when `_owner` sends ether to the MiniMe Token contract
    /// @param _owner The address that sent the ether to create tokens
    /// @return True if the ether is accepted, false if it throws
    function proxyPayment(address _owner, bytes4 sig, bytes data) payable public returns (bool);

    /// @notice Notifies the controller about a token transfer allowing the
    ///  controller to react if desired
```

```
    /// @param   from The origin of the transfer
    /// @param   to The destination of the transfer
    /// @param _amount The amount of the transfer
    /// @return False if the controller does not authorize the transfer
    function onTransfer(address   from, address   to, uint   amount) public returns (bool);

    /// @notice Notifies the controller about an approval allowing the
    ///   controller to react if desired
    /// @param   owner The address that calls `approve()`
    /// @param   spender The spender in the `approve()` call
    /// @param   amount The amount in the `approve()` call
    /// @return False if the controller does not authorize the approval
    function onApprove(address   owner, address   spender, uint   amount) public returns (bool);
}
// Beosin (Chengdu LianAn) // Define interface function.
contract ApproveAndCallFallBack {
    function receiveApproval(address from, uint256   amount, address   token, bytes   data) public;
}
// Beosin (Chengdu LianAn) // Define interface function.
contract ERC223 {
    function transfer(address to, uint amount, bytes data) public returns (bool ok);

    function transferFrom(address from, address to, uint256 amount, bytes data) public returns (bool ok);

    event ERC223Transfer(address indexed from, address indexed to, uint amount, bytes data);
}

contract KTON is DSToken("KTON"), ERC223 {
    address public controller; // Beosin (Chengdu LianAn) // Declare the variable 'controller' for storing the controller address.

    constructor() public {
        setName("Evolution Land Kryptonite");
        controller = msg.sender;
    }

//////////
// Controller Methods
//////////
    /// @notice Changes the controller of the contract
    /// @param _newController The new controller of the contract
    function changeController(address _newController) auth {
        controller = _newController;
    }

    /// @notice Send `_amount` tokens to `_to` from `_from` on the condition it
    ///   is approved by `_from`
    /// @param _from The address holding the tokens being transferred
```

```
    /// @param   to The address of the recipient
    /// @param   amount The amount of tokens to be transferred
    /// @return True if the transfer was successful
    function transferFrom(address   from, address   to, uint256   amount
    ) public returns (bool success) {
        // Alerts the token controller of the transfer
        if (isContract(controller)) { // Beosin (Chengdu LianAn) // If 'controller' is contract address and the result of calling 'onTransfer' function in controller contract is false, this function call will revert.
            if (!TokenController(controller).onTransfer(  from,   to,   amount))
                revert();
        }

        success = super.transferFrom(  from,   to,   amount); // Beosin (Chengdu LianAn) // Execute the function 'transferFrom' of the parent contract to token transfer.
    }

    /*
     * ERC 223
     * Added support for the ERC 223 "tokenFallback" method in a "transfer" function with a payload.
     */
    function transferFrom(address _from, address _to, uint256 _amount, bytes _data)
        public
        returns (bool success)
    {
        // Alerts the token controller of the transfer
        if (isContract(controller)) { // Beosin (Chengdu LianAn) // If 'controller' is contract address and the result of calling 'onTransfer' function in controller contract is false, this function call will revert.
            if (!TokenController(controller).onTransfer(_from, _to, _amount))
                revert();
        }

        require(super.transferFrom(_from, _to, _amount)); // Beosin (Chengdu LianAn) // Execute the function 'transferFrom' of the parent contract to token transfer and check execute result.

        if (isContract(_to)) { // Beosin (Chengdu LianAn) // If '_to' is contract address, call the function 'tokenFallback' of 'receiver' contract.
            ERC223ReceivingContract receiver = ERC223ReceivingContract(_to);
            receiver.tokenFallback(_from, _amount, _data);
        }

        emit ERC223Transfer(_from, _to, _amount, _data); // Beosin (Chengdu LianAn) // Trigger the event 'ERC223Transfer'.s

        return true;
    }

    /*
     * ERC 223
```

```
     * Added support for the ERC 223 "tokenFallback" method in a "transfer" function with a payload.
     * https://github.com/ethereum/EIPs/issues/223
     * function transfer(address _to, uint256 _value, bytes _data) public returns (bool success);
     */
    /// @notice Send `  value` tokens to `  to` from `msg.sender` and trigger
    /// tokenFallback if sender is a contract.
    /// @dev Function that is called when a user or another contract wants to transfer funds.
    /// @param   to Address of token receiver.
    /// @param   amount Number of tokens to transfer.
    /// @param   data Data to be sent to tokenFallback
    /// @return Returns success of function call.
    function transfer(
        address   to,
        uint256   amount,
        bytes   data)
        public
        returns (bool success)
    {
        return transferFrom(msg.sender,   to,   amount,   data);
    }

    /// @notice `msg.sender` approves `_spender` to spend `_amount` tokens on
    ///   its behalf. This is a modified version of the ERC20 approve function
    ///   to be a little bit safer
    /// @param _spender The address of the account able to transfer the tokens
    /// @param _amount The amount of tokens to be approved for transfer
    /// @return True if the approval was successful
    function approve(address _spender, uint256 _amount) returns (bool success) {
        // Alerts the token controller of the approve function call
        if (isContract(controller)) { // Beosin (Chengdu LianAn) // If 'controller' is contract address and the result of calling 'onTransfer' function in controller contract is false, this function call will revert.
            if (!TokenController(controller).onApprove(msg.sender, _spender, _amount))
                revert();
        }

        return super.approve(_spender, _amount); // Beosin (Chengdu LianAn) // Execute the function 'approve' of the parent contract to token approve, return the execute result.
    }

    function mint(address _guy, uint _wad) auth stoppable {
        super.mint(_guy, _wad); // Beosin (Chengdu LianAn) // Execute the function 'mint' of the parent contract to mint tokens.

        emit Transfer(0, _guy, _wad); // Beosin (Chengdu LianAn) // Trigger the event 'Transfer'.
    }
    function burn(address _guy, uint _wad) auth stoppable {
        super.burn(_guy, _wad); // Beosin (Chengdu LianAn) // Execute the function 'burn' of the parent contract to destroy tokens.
```

```
        emit Transfer(  guy, 0,   wad); // Beosin (Chengdu LianAn) // Trigger the event 'Transfer'.
    }

    /// @notice `msg.sender` approves `  spender` to send `  amount` tokens on
    ///   its behalf, and then a function is triggered in the contract that is
    ///   being approved, `  spender`. This allows users to use their tokens to
    ///   interact with contracts in one function call instead of two
    /// @param   spender The address of the contract able to transfer the tokens
    /// @param   amount The amount of tokens to be approved for transfer
    /// @return True if the function call was successful
    function approveAndCall(address   spender, uint256   amount, bytes   extraData
    ) returns (bool success) {
        if (!approve(  spender,   amount)) revert(); // Beosin (Chengdu LianAn) // If approve failed, this function call will revert.

        ApproveAndCallFallBack(  spender).receiveApproval(
            msg.sender,
              amount,
            this,
            _extraData
        ); // Beosin (Chengdu LianAn) // Call the function 'receiveApproval' of '_spender' contract to respond the approval.

        return true;
    }

    /// @dev Internal function to determine if an address is a contract
    /// @param _addr The address being queried
    /// @return True if `_addr` is a contract
    function isContract(address _addr) constant internal returns(bool) {
        uint size;
        if (_addr == 0) return false;
        assembly {
            size := extcodesize(_addr)
        }
        return size>0;
    }

    /// @notice The fallback function: If the contract's controller has not been
    ///   set to 0, then the `proxyPayment` method is called which relays the
    ///   ether and creates tokens as described in the token controller contract
    function ()   payable {
        if (isContract(controller)) { // Beosin (Chengdu LianAn) // If 'controller' is contract address and the result of calling 'proxyPayment' function in controller contract is false, this function call will revert.
            if (! TokenController(controller).proxyPayment.value(msg.value)(msg.sender, msg.sig, msg.data))
                revert();
        } else {
```

```
                revert();
            }
        }

//////////
// Safety Methods
//////////

        /// @notice This method can be used by the owner to extract mistakenly
        ///  sent tokens to this contract.
        /// @param  token The address of the token contract that you want to recover
        ///  set to 0 in case you want to extract ether.
        function claimTokens(address  token) auth {
            if (  token == 0x0) { // Beosin (Chengdu LianAn) // If _token address is equal to zero address, send the all ETH of this contract to the caller.
                address(msg.sender).transfer(address(this).balance);
                return;
            }

            ERC20 token = ERC20(_token);
            uint balance = token.balanceOf(this);
            token.transfer(address(msg.sender), balance); // Beosin (Chengdu LianAn) // Call the function 'transfer' of 'token' contract to send tokens to caller.

            emit ClaimedTokens(_token, address(msg.sender), balance); // Beosin (Chengdu LianAn) // Trigger the event 'ClaimedTokens'.
        }

////////////////
// Events
////////////////

        event ClaimedTokens(address indexed _token, address indexed _controller, uint _amount); // Beosin (Chengdu LianAn) // Declare the events 'ClaimedTokens'.
}
```

成都链安

BEOSIN

Official Website

https://lianantech.com

E-mail

vaas@lianantech.com

Twitter

https://twitter.com/Beosin_com