提案内容

以太坊改进提案EIP223（2017-05-03）定义了代币和代币交互标准，它描述了一种可替代代币的接口，并支持回调tokenReceived函数以在受到代币时通知合约接收者。

该提案为合约引入了一种交互模型，可用于规范与代币交互的合约的行为，该模型包含的内容如下：

1.转移代币时通知接收代币的合约。ERC20中代币转移的接收者不会收到通知。

2.以更节约gas的方式将代币存入合约。

3.允许为代币转移交易附加金融转账操作记录_data。

在本提案中，接收代币的合约必须实现tokenReceived，若代币接收合约没有实现tokenReceived，则必须执行revert。

代币标准API和规范

函数(Methods)

totalSupply

返回代币的总供应量，与ERC20标准的功能相同。

```
function totalSupply() view returns (uint256 totalSupply)
```

name

返回代币的名称，与ERC20标准的功能相同。

为可选项，可以用来提高可用性，但是接口和其它合约不能假定它必须存在。

```
function name() view returns (string _name)
```

symbol

返回代币的符号，与ERC20标准的功能相同。

为可选项，可以用来提高可用性，但是接口和其它合约不能假定它必须存在。

```
function symbol() view returns (string _decimals)
```

decimals

返回代币的精确度，与ERC20标准的功能相同。

为可选项，可以用来提高可用性，但是接口和其它合约不能假定它必须存在。

```
function decimals() view returns (uint8 _decimals)
```

balanceOf

返回给定的owner地址的代币余额，与ERC20标准的功能相同。

```
function balanceOf(address _owner) view returns (uint256 balance)
```

transfer(address,uint)

将value数量的代币转移到地址to，该函数必须调用to地址的tokenReceived(address,uint,bytes calldata)函数。如果to是合约地址且没有实现tokenReceived函数，则代币转移交易必须被中止且相关操作需要被revert；如果to是个人账户地址，则该笔代币转移交易中不执行to的tokenReceived函数。如果消息调用者的账户余额没有足够的代币，则该函数需要抛出异常。在代币交易中可以附加data字段，但这会消耗更多的gas（data字段可以为空）。

注意，即使转移代币的数量为0也必须被视为正常调用并触发Transfer事件。

```
function transfer(address _to, uint256 _value) returns (bool)
```

transfer(address,uint,bytes)

与transfer(address,uint)类似，区别在于该函数可以指定调用tokenReceived函数时的附加字段，即该函数会调用to地址的tokenReceived(address,uint,bytes)函数，而transfer(address,uint)函数调用tokenReceived时一般不附加字段，data置为零值。

我们可以通过查看to地址是否携带code（合约代码）来判断to是否为合约地址。

```
function transfer(address _to, uint256 _value, bytes calldata _data) returns
(bool)
```

事件(Events)

Transfer

代币转移操作需要触发的事件，与ERC20标准的事件相同。

```
event Transfer(address indexed _from, address indexed _to, uint256 _value, bytes _data)
```

ERC-223代币接受合约的接收方法

tokenReceived

当代币持有者发送代币时，代币合约调用的处理代币转移的函数。from为代币发送者的地址，value时转移的代币数量，dara是附加数据。tokenReceived函数的工作原理类似于合约的fallback函数，并且执行时不返回任何内容。

调用tokenReceived函数时msg.sender为对应代币的代币合约，因此tokenReceived可以通过msg.sender分别分辨当前交易发送的是哪个代币，而代币的发送者则是由tokenReceived的from字段标识。

该函数必须被命名为为tokenReceived，且携带的参数必须为address，uint256，bytes，只有这样才能匹配上函数选择器0x8943ec02。

```
function tokenReceived(address _from, uint _value, bytes calldata _data)
```

原理

出于安全考虑，本提案引入了一种代币交互模型，该模型中强制要求在目标合约地址中执行代币接收处理函数。代币接收合约必须显示实现代币处理功能，否则会回滚整个代币转移操作。

本提案中资产转移操作在发送方发起并在接收方处理，因此ERC223的转账在处理将代币存入合约的操作时会更加节约gas，因为ERC223代币只需要一次交易就可以将代币存入合约，而ERC20代币至少需要两次调用（第一次用approve授权，第二次调用transferFrom）。

- ERC-20存款：**approve** ~46 gas，**transferFrom** ~75K gas
- ERC-223存款：**transfer** ~54K gas

ERC20标准实现了两种代币转移方法：①直接使用**transfer**函数；②使用**approve**+**transferFrom**。ERC20的transfer函数不会通知接收者，因此如果代币被发送至其它具有transfer函数的合约，则接收方合约无法识别该代币转移操作，导致代币被永远放置在接收方合约中且无法恢复（根源在于接收方合约没有收到通知，因此无法对自更新自身的合约状态）。

ERC223旨在简化代币的交互流程。ERC223使用"存款"模式对代币进行操作，类似于原生的以太币。对于ERC223合约来说转账就是对transfer函数的简单调用，是一笔单独的交易，而不是ERC20中approve+transferFrom的模式。

ERC标准允许使用**“bytes calldata _data”**参数将有效的数据负载添加在转账交易中，这些负载可以在接收合约中被编码并执行二次调用，类似于msg.data在ether转账交易中的做法；也可以在链上进行公共日志记录或相关金融业务中所必要的操作。

兼容性

本提案的接口类似于ERC20标准，并且大多数功能与ERC20中类似，但是ERC223的**transfer(address, uint256, bytes calldata)**函数不能向后兼容ERC20接口。

ERC20代币可以被转移到具有转账功能的非合约地址或具有approve+transferFrom模式的合约地址，但是若将ERC20代币存入具有transfer函数的合约地址则会导致该笔代币存入操作不被接收方合约认可。

下面是处理ERC20代币存款的合约示例，该合约可以接收tokenA存款，但是无法阻止通过transfer函数将其它代币发送往该合约的操作。

```
contract ERC20Receiver
{
    event Deposit();
    address tokenA; //代币合约地址
    function deposit(uint _value, address _token) public
    {
        require(_token == tokenA);
        IERC20(_token).transferFrom(msg.sender, address(this), _value);
        emit Deposit();
    }
}
```

下面是处理ERC223代币存款的合约示例，该合约可以过滤其它代币仅接收tokenA存款，而其它的ERC223代币存款将会被拒绝。

```
contract ERC223Receiver
{
    event Deposit();
    address tokenA;
    function tokenReceived(address _from, uint _value, bytes memory _data)
public
    {
        require(msg.sender == tokenA);
        emit Deposit();
    }
}
```

安全性

由于ERC223代币模型类似于Ether，因此必须考虑重放问题。

示例

```
pragma solidity ^0.8.19;

library Address {
    /**
     * @dev Returns true if `account` is a contract.
     *
     * This test is non-exhaustive, and there may be false-negatives: during the
     * execution of a contract's constructor, its address will be reported as
     * not containing a contract.
     *
     * > It is unsafe to assume that an address for which this function returns
     * false is an externally-owned account (EOA) and not a contract.
     */
     //判断是否是合约地址（这里用的方法存在风险，合约的构造函数执行指令时extcodesize返回的是0）
    function isContract(address account) internal view returns (bool) {
        // This method relies in extcodesize, which returns 0 for contracts in
        // construction, since the code is only stored at the end of the
        // constructor execution.

        uint256 size;
        // solhint-disable-next-line no-inline-assembly
        assembly { size := extcodesize(account) }
        return size > 0;
    }
}

abstract contract IERC223Recipient {
/**
 * @dev Standard ERC223 function that will handle incoming token transfers.
 *
 * @param _from  Token sender address.
 * @param _value Amount of tokens.
 * @param _data  Transaction metadata.
 */
    function tokenReceived(address _from, uint _value, bytes memory _data) public virtual;
}

/**
 * @title Reference implementation of the ERC223 standard token.
 */
contract ERC223Token {

    /**
     * @dev Event that is fired on successful transfer.
     */
```

```
    event Transfer(address indexed from, address indexed to, uint value, bytes data);

    string  private _name;
    string  private _symbol;
    uint8   private _decimals;
    uint256 private _totalSupply;

    mapping(address => uint256) public balances; // List of user balances.

    /**
     * @dev Sets the values for {name} and {symbol}, initializes {decimals} with
     * a default value of 18.
     *
     * To select a different value for {decimals}, use {_setupDecimals}.
     *
     * All three of these values are immutable: they can only be set once during
     * construction.
     */

    constructor(string memory new_name, string memory new_symbol, uint8 new_decimals)
    {
        _name     = new_name;
        _symbol   = new_symbol;
        _decimals = new_decimals;
    }

    /**
     * @dev Returns the name of the token.
     */
    function name() public view returns (string memory)
    {
        return _name;
    }

    /**
     * @dev Returns the symbol of the token, usually a shorter version of the
     * name.
     */
    function symbol() public view returns (string memory)
    {
        return _symbol;
    }

    /**
     * @dev Returns the number of decimals used to get its user representation.
     * For example, if `decimals` equals `2`, a balance of `505` tokens should
     * be displayed to a user as `5,05` (`505 / 10 ** 2`).
```

```
     *
     * Tokens usually opt for a value of 18, imitating the relationship between
     * Ether and Wei. This is the value {ERC223} uses, unless {_setupDecimals} is
     * called.
     *
     * NOTE: This information is only used for _display_ purposes: it in
     * no way affects any of the arithmetic of the contract, including
     * {IERC223-balanceOf} and {IERC223-transfer}.
     */
    function decimals() public view returns (uint8)
    {
        return _decimals;
    }

    /**
     * @dev See {IERC223-totalSupply}.
     */
    function totalSupply() public view returns (uint256)
    {
        return _totalSupply;
    }



    /**
     * @dev Returns balance of the `_owner`.
     *
     * @param _owner   The address whose balance will be returned.
     * @return balance Balance of the `_owner`.
     */
    function balanceOf(address _owner) public view returns (uint256)
    {
        return balances[_owner];
    }

    /**
     * @dev Transfer the specified amount of tokens to the specified address.
     *      Invokes the `tokenFallback` function if the recipient is a contract.
     *      The token transfer fails if the recipient is a contract
     *      but does not implement the `tokenFallback` function
     *      or the fallback function to receive funds.
     *
     * @param _to    Receiver address.
     * @param _value Amount of tokens that will be transferred.
     * @param _data  Transaction metadata.
     */
    function transfer(address _to, uint _value, bytes calldata _data) public returns (bool success)
    {
```

```
        // Standard function transfer similar to ERC20 transfer with no _data
.
        // Added due to backwards compatibility reasons .
        // 更新余额
        balances[msg.sender] = balances[msg.sender] - _value;
        balances[_to] = balances[_to] + _value;
        if(Address.isContract(_to)) {
            //调用接收合约的tokenReceived对此次转账进行处理
            IERC223Recipient(_to).tokenReceived(msg.sender, _value, _data);
        }
        emit Transfer(msg.sender, _to, _value, _data);
        return true;
    }

    /**
     * @dev Transfer the specified amount of tokens to the specified address.
     *      This function works the same with the previous one
     *      but doesn't contain `_data` param.
     *      Added due to backwards compatibility reasons.
     *
     * @param _to    Receiver address.
     * @param _value Amount of tokens that will be transferred.
     */
    function transfer(address _to, uint _value) public returns (bool success)
    {
        bytes memory _empty = hex"00000000"; //不附带data，置为零值
        balances[msg.sender] = balances[msg.sender] - _value;
        balances[_to] = balances[_to] + _value;
        if(Address.isContract(_to)) {
            IERC223Recipient(_to).tokenReceived(msg.sender, _value, _empty);
        }
        emit Transfer(msg.sender, _to, _value, _empty);
        return true;
    }
}
```

资料来源

ERC-223: 223 Token with communication model (ethereum.org)