简介

EIP165实现了一种标准方法来发布和检测智能合约实现的接口（EIP165，2018-01-23，需要EIP214）。

EIP165标准化了一下内容：

1.如何识别接口?

2.合约如何发布它实现的接口?

3.如何检测合约是否实现了ERC165?

4.如何检测合约是否实现了任何给定的接口?

对于一些类似于ERC20的“标准接口”，有时如果能够通过某种方式查询某些合约是否支持该接口，接口是哪个版本的话，那么我们就可以根据查询结果调整与该合约的交互方式。当前对于ERC20接口来说，以太坊已经提出了适配于它的查询辨别方式，而本提案则是规范了接口的概念和对于接口的识别方法。

标准化

1 如何识别接口

接口是由以太坊ABI定义的一组函数选择器，**interface**关键字定义了返回类型，可变性和时间。接口标识符被定义成了接口中所有函数选择器的异或运算结果，下面的代码展示了如何计算接口标识符。

```
pragma solidity ^0.4.20;

interface Solidity101 {
    function hello() external pure;
    function world(int) external pure;
}

contract Selector {
    function calculateSelector() public pure returns (bytes4) {
        Solidity101 i;
        return i.hello.selector ^ i.world.selector; //接口的所有函数选择器求异或
    }
}
```

calculateSelector得到的结果就是接口标识符。

2 合约如何发布它实现的接口

符合ERC165的合约应该实现下列接口（称为ERC165.sol）。

```solidity
pragma solidity ^0.4.20;

interface ERC165 {
    /// @notice Query if a contract implements an interface
    /// @param interfaceID The interface identifier, as specified in ERC-165
    /// @dev Interface identification is specified in ERC-165. This function
    ///  uses less than 30,000 gas.
    /// @return `true` if the contract implements `interfaceID` and
    ///  `interfaceID` is not 0xffffffff, `false` otherwise
    function supportsInterface(bytes4 interfaceID) external view returns
(bool);
}
```

上面接口的标识符是**0x01ffc9a7**，该标识符可以通过**bytes4(keccak256('supportsInterface(bytes4)'))**方法或使用**Selector**来获得。

当合约实现ERC165接口后supportsInterface函数返回值可能存在的情况如下：

当interfaceID为**0x01ffc9a7**时，返回**true**；

当interfaceID为**0xffffffff**时，但会**false**；

当interfaceID为该合约实现的其它接口标识符时，返回**true**，否则返回**false**。

supportsInterface函数消耗的gas不超过30000。

3 如何检测合约实现了ERC165接口

检测步骤如下：

1.向被检测合约使用STATICCALL，calldata为**0x01ffc9a701ffc9a700000000000000000000000000000000000000000000000000000000**，gas为30000。这相当于调用了被测合约的**supportsInterface(0x01ffc97)**。

2.如果上述调用返回false，则目标合约没有实现ERC-165。

3.如果返回的时true，则进行第二次调用，calldata为**0x01ffc9a7ffffffff00000000000000000000000000000000000000000000000000000000**。

4.如果第二次调用返回true，则被测合约没有实现ERC-165。

5.如果第二次返回false，则被测合约实现了ERC-165。

4 如何检测合约是否实现了任何给定的接口

1.先用上面3中的方法检测被测合约是否实现了ERC165接口。

2.如果被测合约没有实现ERC165，则无法使用ERC165对给定接口进行检测。

3.如果合约实现了ERC165，则调用被测合约的supportsInterface检测即可。

测试用例

1.查询合约

```
pragma solidity ^0.4.20;

contract ERC165Query {
    bytes4 constant InvalidID = 0xffffffff;
    bytes4 constant ERC165ID = 0x01ffc9a7;

    function doesContractImplementInterface(address _contract, bytes4 _interfaceId) external view returns (bool) {
        uint256 success;
        uint256 result;

        (success, result) = noThrowCall(_contract, ERC165ID);
        if ((success==0)||(result==0)) {
            return false;
        }

        (success, result) = noThrowCall(_contract, InvalidID);
        if ((success==0)||(result!=0)) {
            return false;
        }

        (success, result) = noThrowCall(_contract, _interfaceId);
        if ((success==1)&&(result==1)) {
            return true;
        }
        return false;
    }

    function noThrowCall(address _contract, bytes4 _interfaceId) constant internal returns (uint256 success, uint256 result) {
        bytes4 erc165ID = ERC165ID;
        //使用内联汇编
        assembly {
            //获取下一个可用的存储槽
            let x := mload(0x40)               // Find empty storage location using "free memory pointer"
            //设置函数选择器，选择目标合约的supportsInterface函数
            mstore(x, erc165ID)                // Place signature at beginning of empty storage
            //设置传入参数，为接口ID，与函数选择器拼接在一起
            mstore(add(x, 0x04), _interfaceId) // Place first argument directly next to signature
            //success为true则代表目标合约实现了supportsInterface函数
            success := staticcall(
                30000,          // 30k gas
                _contract,      // To addr
                x,              // Inputs are stored at location x
                0x24,           // Inputs are 36 bytes long
                x,              // Store output over input (saves space)
```

```
                0x20)          // Outputs are 32 bytes long
                //result为1则代表supportsInterface返回true
                result := mload(x)                // Load the result
        }
    }
}
```

2.实现ERC165接口的合约

下列实现方式中supportsInterface的执行开销是586gas，但是ERC165MappingImplementation合约的部署和supportedInterfaces的存储需要消耗额外的gas（ERC165MappingImplementation可以重复使用）。

```
pragma solidity ^0.4.20;

import "./ERC165.sol";

contract ERC165MappingImplementation is ERC165 {
    /// @dev You must not set element 0xffffffff to true
    mapping(bytes4 => bool) internal supportedInterfaces;

    function ERC165MappingImplementation() internal {
        supportedInterfaces[this.supportsInterface.selector] = true;
    }

    function supportsInterface(bytes4 interfaceID) external view returns (bool)
{
        return supportedInterfaces[interfaceID];
    }
}

interface Simpson {
    function is2D() external returns (bool);
    function skinColor() external returns (string);
}

contract Lisa is ERC165MappingImplementation, Simpson {
    function Lisa() public {
        supportedInterfaces[this.is2D.selector ^ this.skinColor.selector] =
true;
    }

    function is2D() external returns (bool){}
    function skinColor() external returns (string){}
}
```

下面是第二种实现方式,supportsInterface的返回值通过计算实现，最差情况下的执行开销为236个gas，但随着合约支持的接口数量增加开销也会线性增加。这种方式没有对0xffffffff进行判断，因此在查询时需要额外处理。

```
```

```solidity
pragma solidity ^0.4.20;

import "./ERC165.sol";

interface Simpson {
    function is2D() external returns (bool);
    function skinColor() external returns (string);
}

contract Homer is ERC165, Simpson {
    function supportsInterface(bytes4 interfaceID) external view returns (bool)
{
        return
            //当实现ERC165接口是返回true
          interfaceID == this.supportsInterface.selector || // ERC165
          interfaceID == this.is2D.selector //当实现Simpson接口时返回true
                         ^ this.skinColor.selector; // Simpson
    }

    function is2D() external returns (bool){}
    function skinColor() external returns (string){}
}
```

当一个合约支持超过三个或以上数量的接口时（包括ERC165接口本身），使用第一种实现方式调用supportsInterface函数进行查询时消耗的gas更少。

资料来源

ERC-165: Standard Interface Detection (ethereum.org)