**矩阵元区块链**

**智能合约开发指南**

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本号： | 1.0.0 | 文档编号： |  |
| 文档密级： |  | 归属部门/项目： |  |
| 产品名： | Juzix\_BlockChain | 子系统名： |  |
| 编写人： |  | 编写日期： |  |

****

**修订记录:**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本号** | **修订人** | **修订日期** | **修订内容** |
| V1.0.0 | 区块链项目组 | 2017.7.5 | 矩阵元区块链智能合约开发指南 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**所有权声明**

除特别声明外，此文档所用的公司名称、个人姓名及数据均属为说明的目的而模拟。

本文档的版权属矩阵元技术(深圳)有限公司所有，受中华人民共和国法律的保护。

本文档所含的任何构思、设计、工艺及其他技术信息均属本公司所有，受中华人民共和国法律的保护。未经本公司书面同意，任何单位和个人不得擅自摘抄、全部或部分复制本书内容，或者以其他任何方式使第三方知悉。

除非另有约定，本手册仅作为使用指导，本手册中的所有陈述、信息和建议不构成任何明示或暗示的担保。由于产品版本升级或其它原因，本手册内容会不定期更新，恕不另行通知。

目录

[1. 适用范围 4](#_Toc2043)

[2. 术语解释 4](#_Toc24149)

[3. Quick Start 4](#_Toc2517)

[3.1. 合约功能描述 4](#_Toc17243)

[3.2. 合约规划与列表说明 4](#_Toc19369)

[3.2.1. 合约存储规划 4](#_Toc6479)

[3.2.2. 合约接口规划 5](#_Toc11545)

[3.2.3. 合约权限规划 5](#_Toc10809)

[3.2.4. 合约文件规划 5](#_Toc15698)

[3.3. 合约代码样例 5](#_Toc2247)

[3.4. 合约编译发布 11](#_Toc27550)

[3.4.1. 前提要求 11](#_Toc14630)

[3.4.2. 合约编译 11](#_Toc23891)

[3.4.3. 合约发布 12](#_Toc15795)

[3.4.4. 合约测试 12](#_Toc7246)

[4. 系统合约接口说明 12](#_Toc3578)

[4.1. Library库接口说明 12](#_Toc10862)

[4.1.1. LibString.sol 12](#_Toc29404)

[4.1.2. LibInt.sol 17](#_Toc10654)

[4.1.3. LibJson.sol 18](#_Toc22753)

[4.2. 系统合约接口说明 19](#_Toc1392)

[4.2.1. OwnerNamed.sol 19](#_Toc29168)

[4.2.2. RegisterManager.sol 21](#_Toc12889)

[4.2.3. UserManager.sol 22](#_Toc2574)

[4.2.4. RoleManager.sol 27](#_Toc30736)

[4.2.5. DepartmentManager.sol 30](#_Toc20179)

[4.2.6. ActionManager.sol 33](#_Toc10143)

[4.3. 系统角色与权限说明 35](#_Toc1966)

[5.合约业务开发流程介绍 35](#_Toc25371)

[5.1完成合约基础数据的处理 36](#_Toc27959)

[5.1.1 string型数据的截取： 36](#_Toc3117)

[5.1.2 int型数据到string型数据的转换： 36](#_Toc27)

[5.2实现注册合约 37](#_Toc22478)

[5.3定义业务数据结构 38](#_Toc1132)

[5.3.1组织合约，组织的数据结构： 38](#_Toc14559)

[5.3.2角色合约，角色的数据结构： 39](#_Toc11347)

[5.3.3用户合约，用户的数据结构： 39](#_Toc8323)

[5.3.4权限合约，权限的数据结构： 40](#_Toc5497)

[5.4根据接口文档实现业务合约的业务逻辑 40](#_Toc19058)

[5.5模块化管理 41](#_Toc5800)

# 适用范围

本规范描述了矩阵元区块链系统智能合约的开发约束与规范，用以指导DAPP开发者按照本规范开发基于矩阵元区块链运行的应用。

# 术语解释

|  |  |
| --- | --- |
| 术语 | 术语解释 |
| DAPP | 去中心化应用 |
| Truffle | 智能合约开发IDE |

# Quick Start

本章节描述一个简单的智能合约开发样例，用以描述基于矩阵元区块链的智能合约开发标准与规范。给DAPP应用的开发提供参考。

## 合约功能描述

此合约开发用例用来管理学生数据，并给不同用户分配不同的数据处理权限，实现数据

的访问控制。

## 合约规划与列表说明

### 合约存储规划

学生数据在合约的存储属性为：

|  |  |
| --- | --- |
| id | 学生Id（对应于钱包地址） |
| name | 学生名字 |
| classId | 学生所属班级id |
| status | 标记学生数据是否开除： 0 未开除 1 已开除 |

### 合约接口规划

//新增学生

func addStudent(string \_studentJson) public returns{};

//开除学生

func deleteStudentById0(string id) public{};

//根据Id查询用户

func findStudentById(string id) constant public returns{};

### 合约权限规划

|  |  |
| --- | --- |
| 角色列表 | 角色具有的权限 |
| 校长 | deleteStudentById, addStudent, findStudentById |
| 班主任 | addStudent, findStudentById |
| 学生 | findStudentById |

### 合约文件规划

* LibStudent.sol: 学生的数据结构合约；
* StudentManager.sol: 实现对学生数据的各种处理；
* Module.sol: 实现对合约、角色、权限的模块化处理。

## 合约代码样例

* LibStudent.sol:

pragma solidity ^0.4.2;

import "LibInt.sol";

import "LibString.sol";

import "LibJson.sol";

library LibStudent {

using LibJson for \*;

using LibString for \*;

using LibInt for \*;

using LibStudent for \*; //引入库的语法

struct Student{

address id; //学生Id

string name; //学生名称

string classId; //学生班级Id

uint status; //学生状态 0 未开除 1 开除

} //student对象结构体

//将student对象转换成json字符串

function toJson(Student storage \_self) internal returns(string \_strjson){

\_strjson = "{";

string memory strAddr = "0x";

strAddr = strAddr.concat(\_self.id.addrToAsciiString());

\_strjson = \_strjson.concat(strAddr.toKeyValue("id"), ",")

\_strjson = \_strjson.concat(\_self.name.toKeyValue("name"), ",");

\_strjson = \_strjson.concat(\_self.classId.toKeyValue("classId"), ",");

\_strjson = \_strjson.concat(uint(\_self.status).toKeyValue("status"), " ");

\_strjson = \_strjson.concat("}");

}

//从student的json字符串中提取相应值

function fromJson(Student storage \_self, string \_stuJson) internal returns(bool) {

\_self.clear();

string memory strAddr = \_stuJson.getStringValueByKey("id");

strAddr = \_stuJson.getStringValueByKey("id");

\_self.id= strAddr.toAddress();

\_self.name = \_stuJson.jsonRead("name");

\_self.classId= \_stuJson.jsonRead("classId");

\_self.status = \_stuJson.jsonRead("status").toUint();

return true;

}

//将student对象置空

function clear(Student storage \_self) internal {

\_self.id= address(0);

\_self.name = "";

\_self.classId= "";

\_self.status = 1;

}

//重置student对象

function reset(Student storage \_self) internal {

clear(\_self);

\_self.clear();

}

}

StudentManager.sol:

pragma solidity ^0.4.2;

import "LibStudent.sol";

import "LibString.sol";

import "LibInt.sol";

contract StudentManager {

using LibStudent for \*;

using LibString for \*;

using LibInt for \*; //引入所需库合约

event Notify(uint \_errno, string \_info); //定义事件处理

//此mapping对象将地址映射到对象本身

mapping(address=>LibStudent.Student) studentMap;

address[] addrList;

address[] tempList;

LibStudent.Student internal student;

string[] tmpArray;

enum StudentError {

NO\_ERROR,

BAD\_PARAMETER,

NAME\_EMPTY,

CLASS\_NOT\_EXISTS,

STUDENT\_NOT\_EXISTS

}//错误码定义

uint errno = 0;

function StudentManager() {

}//构造函数

//新增学生

function addStudent(string \_studentJson) public returns(uint) {

log("insert", "StudentManager");

if (student.fromJson(\_studentJson) == false) {

log("insert bad json", "StudentManager");

errno = 9000 + uint(StudentError.BAD\_PARAMETER);

Notify(errno, "insert bad json");

return errno;

}

if (student.name.equals("")) {

log("student name is invalid", "StudentManager");

errno = 9000 + uint(StudentError.NAME\_EMPTY);

Notify(errno, "student name is invalid");

return errno;

}

if (student.classId.equals("")) {

log("student classId is invalid", "StudentManager");

errno = 9000 + uint(StudentError.CLASS\_NOT\_EXISTS);

Notify(errno, "student classId is invalid");

return errno;

}

student.status = uint(0);

studentMap[student.id] = student;

addrList.push(student.id);

errno = uint(StudentError.NO\_ERROR);

log("add a student succcess", "StudentManager");

Notify(errno, "insert a student succcess");

student.reset();

return errno;

}

//根据学生Id查找学生

function findStudentById(address \_id) constant public returns(string \_ret) {

\_ret = "{\"ret\":0,\"data\":{";

if (studentMap[\_id].status != 1) {

\_ret = \_ret.concat(studentMap[\_id].toJson());

} else {

\_ret = \_ret.concat(""));

}

\_ret = \_ret.concat("}}");

}

//根据学生Id删除学生

function deleteStudentById(address \_id) public {

if (studentMap[\_id].status == 1) {

log("student not exists: ", \_id);

errno = 9000 + uint(StudentError.STUDENT\_NOT\_EXISTS);

Notify(errno, "student not exists");

return;

}

delete tempList;

for (uint i = 0; i < addrList.length; i++) {

if(\_id == studentMap[addrList[i]].id) {

continue;

}

else {

tempList.push(addrList[i]);

}

}

delete addrList;

for (uint j = 0; j < tempList.length; ++j) {

addrList.push(tempList[j]);

}

studentMap[\_id].status = 1;

Notify(errno, "fire student success");

}

}

ModuleManager.sol:

pragma solidity ^0.4.2;

import "LibModule.sol";

import "LibContract.sol";

import “BaseModule.sol”;

contract ModuleManager is BaseModule{

//构造module对象

LibModule.Module tmpModule;

//构造contract对象

LibContract.Contract tmpContract;

//构造函数，发布入链即自动执行

func ModuleManager(){

tmpModule = LibModule.Module({moduleId:"systemModule001",moduleName:"系统模块",moduleVersion:"0.0.1.0",deleted:false,moduleEnable:1,moduleDescription:"系统模块管理",moduleCreateTime:nowTime,moduleUpdateTime:nowTime,moduleCreator:msg.sender,contractIdList:tmpArr01,roleIds:tmpArr01});

addModule(tmpModule); //添加此模块

initContractData(); //添加合约

initActionData(); //添加权限

initRoleData(); //添加角色

}

//添加合约数据

func initContractData() private returns(uint){

address userAddr = rm.getContractAddress("UserManager","0.0.1.0");

tmpContract.moduleId = sysModuleId;

tmpContract.cctId = innerContractMapping["UserManager"];

tmpContract.cctName = "用户管理";

tmpContract.cctVersion = "0.0.1.0";

tmpContract.deleted = false;

tmpContract.enable = 1;

tmpContract.description = "用户管理合约";

tmpContract.createTime = nowTime;

tmpContract.updateTime = nowTime;

tmpContract.creator = msg.sender;

tmpContract.cctAddr = userAddr;

addContract(tmpContract.toJson());

return 1;

}

//添加权限数据

func initActionData() private returns(uint){

string memory actionStr = "";

//权限数据对象 jsonStr="{\"id\":\"action1000\",\"moduleId\":\"module001\",\"name\":\"addStudent\",\"resKey\":\"StudentManager\",\"opKey\":\"addStudent(string)\"}";

addAction(jsonStr);

jsonStr="{\"id\":\"action1001\",\"moduleId\":\"module001\",\"name\":\"findStudentById\",\"resKey\":\"StudentManager\",\"opKey\":\"findStudentById(string)\"}";

addAction(jsonStr);

jsonStr="{\"id\":\"action1002\",\"moduleId\":\"module001\",\"name\":\"deleteStudentById\",\"resKey\":\"StudentManager\",\"opKey\":\"deleteStudentById(string)\"}";

addAction(jsonStr);

return 1;

}

//添加角色数据

func initRoleData() private returns(uint){

string memory roleJsonStr="";

//角色数据对象

roleJsonStr="{\"id\":\"role1000\",\"name\":\"校长\",\"status\":1,\"moduleId\":\"systemModule001\",\"actionIdList\":[\"action1000\",\"action1001\",\"action1002\"]}";

addRole(roleJsonStr);

roleJsonStr="{\"id\":\"role1001\",\"name\":\"班主任\",\"status\":1,\"moduleId\":\"systemModule001\",\"actionIdList\":[\"action1000\",\"action1001\"]}";

addRole(roleJsonStr);

roleJsonStr="{\"id\":\"role1002\",\"name\":\"学生\",\"status\":1,\"moduleId\":\"systemModule001\",\"actionIdList\":[\"action1001\",]}";

addRole(roleJsonStr);

return 1;

}

}

## 合约编译发布

### 前提要求

* 本地矩阵元区块链平台已经启动运行。

矩阵元区块链平台搭建请参考《矩阵元区块链环境安装部署》文档；

* 本地已搭建好Truffle环境。

Truffle编译发布环境搭建请参考《矩阵元区块链Truffle环境搭建》文档。

### 合约编译与发布

1. Truffle环境初始化

* 新建合约目录DApp
* 在新建的合约目录DApp下执行truffle init初始化目录结构，得到如下的目录树
* 修改truffle.js文件中RPC地址为平台地址
* 删除示例合约ConvertLib.sol，MetaCoin.sol，Migrations.sol

1. 上传应用智能合约

* 将编写的上述智能合约上传到truffle init生成目录结构下的contract中

1. 合约编译

* 在新建的Dapp目录下执行命令truffle compile完成合约的编译

1. 合约发布

* 修改部署配置文件：

在migrations目录下修改migrations/1\_initial\_migration.js

注释掉deployer.deploy(Migrations);

在migrations目录下修改migrations/2\_deploy\_contracts.js

添加自己要发布的合约，例：

module.exports = function(deployer) {

deployer.deploy(LibInt);

deployer.deploy(LibString);

deployer.deploy(LibRole);

deployer.deploy(RoleManager);

};

* 在新建的Dapp目录下执行命令truffle migrate完成合约的发布。

### 合约测试

* 进入truffle控台： truffle console
* 实例化待测合约对象，如StudentManager.sol

> var stu = StudentManager.deployed();

undefined

或者

> var stu = StudentManager.at(“0x002b4f09741a896e757f276d8f5f0c24bca870bf”);

undefined

注：该处此地址为合约发布时输出的对应地址

* 调用合约函数：

> stu.addStudent(‘{“id”: “0x006d8f5f0c24bca870bf2b4f09741a896e757f27”, “name”: “juzix”, “classId”: “SuperClass”,“status”: 0}’);

“0x00aa1676c77bb7861ef115652715671880aa1677”

> stu.findStudentById(“0x006d8f5f0c24bca870bf2b4f09741a896e757f27”);

‘{“id”: “0x006d8f5f0c24bca870bf2b4f09741a896e757f27”, “name”: “juzix”, “classId”: “SuperClass”,“status”: 0}’

> stu.deleteStudentById(“0x006d8f5f0c24bca870bf2b4f09741a896e757f27”);

“0x5652715671880aa00aa1676c77bb7861ef11ba21”

# 系统合约接口说明

## Library库接口说明

### LibString.sol

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function memcpy(uint dest, uint src, uint len) | | |
| 函数描述 | 字符串拷贝 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | dest | uint | 复制对象 |
| src | uint | 被复制对象 |
| len | uint | 复制长度 |
| 返回值 | 无 |  | uint型数据复制 |
| 接口原型 | function compare(string \_self, string \_str) internal returns (int8 \_ret) | | |
| 函数描述 | 字符串比较（区分大小写） | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 比较的第一个对象 |
| \_str | string | 比较的第二个对象 |
| 返回值 | \_ret | Int8 | -1 前比后小 0 相等 1 前比后大 |
| 接口原型 | function compareNoCase(string \_self, string \_str) internal returns (int8 \_ret) | | |
| 函数描述 | 比较两字符串大小（不区分大小写） | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 比较的第一个对象 |
| \_str | string | 比较的第二个对象 |
| 返回值 | \_ret | Int8 | -1 前比后小 0 相等 1 前比后大 |
| 接口原型 | function equals(string \_self, string \_str) internal returns (bool \_ret) | | |
| 函数描述 | 判断两字符串是否相等（区分大小写） | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 第一个对象 |
| \_str | string | 第二个对象 |
| 返回值 | \_ret | bool | True 相等 false 不等 |
| 接口原型 | function equalsNoCase(string \_self, string \_str) internal returns (bool \_ret) | | |
| 函数描述 | 判断两字符串是否相等（不区分大小写） | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 第一个对象 |
| \_str | string | 第二个对象 |
| 返回值 | \_ret | bool | True 相等 false 不等 |
| 接口原型 | function substr(string \_self, uint \_start, uint \_len) internal returns (string \_ret) | | |
| 函数描述 | 字符串截取 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 取值对象 |
| \_start | uint | 开始长度 |
| \_len | uint | 结束长度 |
| 返回值 | \_ret | string | 截取后的子字符串 |
| 接口原型 | function concat(string \_self, string \_str) internal returns (string \_ret) | | |
| 函数描述 | 两字符串拼接 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 第一个对象 |
| \_str | string | 第二个对象 |
| 返回值 | \_ret | string | 拼接后的字符串 |
| 接口原型 | function concat(string \_self, string \_str1, string \_str2) internal returns (string \_ret) | | |
| 函数描述 | 三字符串拼接 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 第一个对象 |
| \_str1 | string | 第二个对象 |
| \_str2 | string | 第三个对象 |
| 返回值 | \_ret | string | 拼接后的字符串 |
| 接口原型 | function concat(string \_self, string \_str1, string \_str2, string \_str3) internal returns (string \_ret) | | |
| 函数描述 | 四字符串拼接 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | uint | 第一个对象 |
| \_str1 | uint | 第二个对象 |
| \_str2 | uint | 第三个对象 |
| \_str3 | uint | 第四个对象 |
| 返回值 | \_ret | string | 拼接后的字符串 |
| 接口原型 | function trim(string \_self) internal returns (string \_ret) | | |
| 函数描述 | 首尾删除空白 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 被操作字符串 |
| 返回值 | \_ret | string | 返回的字符串 |
| 接口原型 | function trim(string \_self, string \_chars) internal returns (string \_ret) | | |
| 函数描述 | 删除具体字符 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 被操作字符串 |
| \_char | string | 被删除的字符 |
| 返回值 | \_ret | string | 返回的字符串 |
| 接口原型 | function indexOf(string \_self, string \_str) internal returns (int \_ret) | | |
| 函数描述 | 在源字符串取具体字符串的索引号 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源字符串 |
| \_str | string | 待取字符串 |
| 返回值 | \_ret | int | 返回的索引号 |
| 接口原型 | function indexOf(string \_self, string \_str, uint pos) internal returns (int \_ret) | | |
| 函数描述 | 在源字符串取具体字符串的索引号 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源字符串 |
| \_str | string | 待取字符串 |
| Pos | uint | 待取字符串的指定起始数 |
| 返回值 | \_ret | int | 返回的索引号 |
| 接口原型 | function toInt(string \_self) internal returns (int \_ret) | | |
| 函数描述 | 字符串型数据转int型 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源字符串 |
| 返回值 | \_ret | int | 返回的int型数据 |
| 接口原型 | function fromHexChar(bytes1 \_i) internal returns (int8 \_ret) | | |
| 函数描述 | 将16进制字节转int8 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_i | bytes1 | 待转字节 |
| 返回值 | \_ret | int | 返回的int数据 |
| 接口原型 | function toHex(string \_self) internal returns (bytes \_ret) | | |
| 函数描述 | 字符串转字节 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源字符串 |
| 返回值 | \_ret | bytes | 返回的字节 |
| 接口原型 | function toAddress(string \_self) internal returns (address \_ret) | | |
| 函数描述 | 字符串转地址 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源字符串 |
| 返回值 | \_ret | address | 返回的地址 |
| 接口原型 | function toKeyValue(string \_self, string \_key) internal returns (string \_ret) | | |
| 函数描述 | 输出key：value型 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | String型value |
| \_key | string | key |
| 返回值 | \_ret | string | 返回key+value 如：”key”:”hello” |
| 接口原型 | function toKeyValue(string[] storage \_self, string \_key) internal returns (string \_ret) | | |
| 函数描述 | 输出key:value型 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string[] | String[]型数据对象 |
| \_key | string | Key |
| 返回值 | \_ret | string | 返回key+value 如： “key”:[“a”,”b”] |
| 接口原型 | function getStringValueByKey(string \_self, string \_key) internal returns (string \_ret) | | |
| 函数描述 | 获得对应的string型value | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源数据对象 |
| \_key | string | Key |
| 返回值 | \_ret | string | 返回对应的value |
| 接口原型 | function getIntValueByKey(string \_self, string \_key) internal returns (int \_ret) | | |
| 函数描述 | 获得对应的int型value | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源数据对象 |
| \_key | string | Key |
| 返回值 | \_ret | int | 返回对应的value |
| 接口原型 | function getArrayValueByKey(string \_self, string \_key) internal returns (string \_ret) | | |
| 函数描述 | 获得对应的array型value | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源数据对象 |
| \_key | string | Key |
| 返回值 | \_ret | string | 返回对应的value |
| 接口原型 | function keyExists(string \_self, string \_key) internal returns (bool \_ret) | | |
| 函数描述 | 判断key是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源数据对象 |
| \_key | string | Key |
| 返回值 | \_ret | bool | False 不存在 true 存在 |
| 接口原型 | function storageToUint(string \_self) internal returns (uint \_ret) | | |
| 函数描述 | String型数据转uint | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源数据对象 |
| 返回值 | \_ret | uint | 返回uint型数据 |
| 接口原型 | function inArray(string \_self, string[] storage \_array) internal returns (bool \_ret) | | |
| 函数描述 | 判断array型数据(区分大小写)是否在源数据对象是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源数据对象 |
| \_array | string[] | 指定的array对象 |
| 返回值 | \_ret | bool | false 不存在 true 存在 |
| 接口原型 | function inArrayNoCase(string \_self, string[] storage \_array) internal returns (bool \_ret) | | |
| 函数描述 | 判断array型数据(不区分大小写)是否在源数据对象是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源数据对象 |
| \_array | string[] | 指定的array对象 |
| 返回值 | \_ret | bool | false 不存在 true 存在 |
| 接口原型 | function addrToAsciiString(address x) internal returns (string) | | |
| 函数描述 | 地址型数据转字符串 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | X | address | 地址型数据 |
| 返回值 | \_ret | string | 返回的字符串 |
| 接口原型 | function toChar(byte b) internal returns (byte c) | | |
| 函数描述 | 字节转字符 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | b | byte | 源字节 |
| 返回值 | c | byte | 返回的字符 |
| 接口原型 | function toUpper(string \_self) internal returns (string \_ret) | | |
| 函数描述 | 字符串变大写 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源字符串 |
| 返回值 | \_ret | string | 返回的字符串 |
| 接口原型 | function toLower(string \_self) internal returns (string \_ret) | | |
| 函数描述 | 字符串小写 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | string | 源字符串 |
| 返回值 | \_ret | string | 返回的字符串 |

### LibInt.sol

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function toString(uint \_self, uint width) internal returns (string \_ret) | | |
| 函数描述 | Uint型转字符串 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | uint | Uint数据对象 |
| width | uint | 待转换长度 |
| 返回值 | \_ret | string | 返回的字符串 |
| 接口原型 | function toString(uint \_self) internal returns (string \_ret) | | |
| 函数描述 | Uint型转字符串 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | uint | Uint数据对象 |
| 返回值 | \_ret | string | 返回的字符串 |
| 接口原型 | function toHexString(uint \_self) internal returns (string \_ret) | | |
| 函数描述 | Uint转16进制字节码 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | uint | Uint数据对象 |
| 返回值 | \_ret | string | 返回的16进制字节码 |
| 接口原型 | function toHexString64(uint \_self) internal returns (string \_ret) | | |
| 函数描述 | Uint转”0x”开头的16进制字节码 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | uint | Uint数据对象 |
| 返回值 | \_ret | string | 返回的以”0x”开头的16进制字节码 |
| 接口原型 | function toString(int \_self) internal returns (string \_ret) | | |
| 函数描述 | Int转字符串 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
|  | \_self | int | int数据对象 |
| 返回值 | \_ret | string | 返回的字符串 |
| 接口原型 | function toAddrString(uint \_self) internal returns (string \_ret) | | |
| 函数描述 | uint型转地址型 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | uint | Uint数据对象 |
| 返回值 | \_ret | string | 返回地址型字符串 |
| 接口原型 | function toKeyValue(uint \_self, string \_key) internal returns (string \_ret) | | |
| 函数描述 | 返回key:value型（value为uint） | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | uint | Uint型value |
| \_key | string | Key |
| 返回值 | \_ret | string | 返回key+value 如 “key”:1 |
| 接口原型 | function toKeyValue(int \_self, string \_key) internal returns (string \_ret) | | |
| 函数描述 | 返回key:value型（value为int） | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | int | int型value |
| \_key | string | key |
| 返回值 | \_ret | string | 返回key+value 如 ”key”:1 |
| 接口原型 | function toKeyValue(address \_self, string \_key) internal returns (string \_ret) | | |
| 函数描述 | 返回key:value型（value为address） | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | address | address型value |
| \_key | string | key |
| 返回值 | \_ret | string | 返回key+value 如 “key”:”0x00ad201d3f88e4991728215” |
| 接口原型 | function recoveryToString(uint \_self) internal returns (string \_ret) | | |
| 函数描述 | uint转字符串 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | uint | uin数据对象 |
| 返回值 | \_ret | string | Uint型转string型 |

### LibJson.sol

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function isJson(string \_json) internal constant returns(bool \_ret) | | |
| 函数描述 | 判断string对象是否json | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | json对象 |
| 返回值 | \_ret | bool | 是否json |
| 接口原型 | jsonRead(string \_json, string \_keyPath) internal constant returns(string \_ret) | | |
| 函数描述 | 获得key对应的value | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | Json对象 |
| \_keyPath | string | 所读取的key |
| 返回值 | \_ret | string | Key对应的value |
| 接口原型 | function jsonKeyExists(string \_json, string \_keyPath) internal constant returns(bool \_ret) | | |
| 函数描述 | 判断key是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | Json对象 |
| \_keyPath | string | 所读取的key |
| 返回值 | \_ret | bool | Key是否存在 |

### LibPaillier.sol

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function pai\_add(string d1, string d2) internal constant returns (string result) | | |
| 函数描述 | 加法同态计算 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_d1 | string | 第一个对象 |
| \_d2 | string | 第二个对象 |
| 返回值 | result | string | 同态相加返回结果 |

### LibContract.sol

合约对象的数据结构：

struct Contract{

string moduleId; //合约所在模块id

string contractId; //合约id

string contractName; //合约名称

string contractVersion; //合约版本

bool deleted; //合约是否删除

uint enable; //合约是否启用

string description; //合约描述

uint createTime; //合约创建时间

uint updateTime; //合约更新时间

address creator; //合约创建者

address contractAddr; //合约地址

}

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function toJson(Contract storage \_self) internal constant returns (string \_json) | | |
| 函数描述 | 将contract对象转换成json字符串 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | Contract | 合约对象 |
| 返回值 | \_json | string | 返回的json字符串 |
| 接口原型 | function fromJson(Contract storage \_self, string \_json) internal constant returns(bool succ) | | |
| 函数描述 | 从contract的json字符串中提取相应字段的值 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | Contract | 合约对象 |
| \_json | string | 合约的json字符串 |
| 返回值 | succ | bool | 是否取值成功 |

### 4.1.6.LibModule.sol

模块对象的数据结构：

struct Module{

string moduleId; //模块id

string moduleName; //模块名称

string moduleVersion; //模块版本

bool deleted; //模块是否删除

uint moduleEnable; //模块开关：0 false 1 true

string moduleDescription; //模块描述

uint moduleCreateTime; //模块创建时间

uint moduleUpdateTime; //模块更新时间

address moduleCreator; //模块创建者

string[] contractIdList; //模块下的合约列表

string[] roleIds; //模块下的角色列表

}

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function toJson(Module storage \_self) constant internal returns(string \_json) | | |
| 函数描述 | 将Module对象转换成json字符串 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | Module | 模块对象 |
| 返回值 | \_json | string | 返回的json字符串 |
| 接口原型 | function fromJson(Module storage \_self, string \_json) constant internal returns(bool succ) | | |
| 函数描述 | 从Module的json字符串中提取相应字段的值 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_self | Module | 模块对象 |
| \_json | string | 模块的json字符串 |
| 返回值 | succ | bool | 是否取值成功 |

### 4.1.6.LibNIZK.sol

零知识证明参数的数据结构：

struct NizkParam{

string      cipher1;          //第一个加数/减数（密文）

string      cipher2;      //第二个加数/减数（密文）

string      pais;               //证明的加密密文

string      balapubcipher;      //账户余额的密文

string      traapubcipher;      //转出账户需要转出的金额密文

address      trabpubcipher;      //转入账户需要转入的金额密文

address      apukkey;      //转出账户的公钥

address      bpukkey;      //转入账户的公钥

address      nizkpp;      //零知识证明的全局结构

}

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function nizk\_setup() internal constant returns (string) | | |
| 函数描述 | 全局一次调用生成链对应的零知识证明结构 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | 无 | string | 返回的零知识证明结构 |
| 接口原型 | function nizk\_apubcipheradd(LibNizkParam.NizkParam param) internal returns (string) | | |
| 函数描述 | 零知识证明下的同态加法 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | param | NizkParam | 零知识证明入参结构 |
| 返回值 | 无 | string | 返回相加后的密文 |
| 接口原型 | function nizk\_apubciphersub(LibNizkParam.NizkParam param) internal returns (string) | | |
| 函数描述 | 零知识证明下的同态减法 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | param | NizkParam | 零知识证明入参结构 |
| 无 | string | 返回相减后的密文 |
| 接口原型 | function nizk\_verifyproof(string pais, string balapubcipher, string traapubcipher, string trabpubcipher, string apukkey, string bpukkey, string nizkpp) internal returns (uint) | | |
| 函数描述 | 验证零知识证明 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | pais | string | 证明的加密密文 |
| balapubcipher | string | 账户余额的密文 |
| traapubcipher | string | 转出账户需要转入的金额密文 |
| trabpubcipher | string | 转入账户需要转入的金额密文 |
| apukkey | string | 转出账户的公钥 |
| bpukkey | string | 转入账户的公钥 |
| nizkpp | string | 零知识证明的全局结构 |
| 返回值 | 无 | uint | 验证结果,验证通过为1 |

## 系统合约接口说明

### OwnerNamed.sol

此合主要提供一些日志输出函数，所有业务合约均继承此合约。

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function register(string \_name, string \_version) public | | |
| 函数描述 | 合约注册 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_\_name | string | 合约名称 |
| \_version | string | 合约版本 |
| 返回值 | 无 |  | 合约注册 |
| 接口原型 | function kill() public | | |
| 函数描述 | 注销合约 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | 无 |  | 注销合约 |
| 接口原型 | function getOwner() constant public returns (string \_ret) | | |
| 函数描述 | 获得合约地址 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_ret | string | 返回合约地址 |
| 接口原型 | function getSender() constant public returns (string \_ret) | | |
| 函数描述 | 获得合约地址 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_无 |  |  |
| 返回值 | \_ret | String | 返回合约地址 |
| 接口原型 | function getErrno() constant returns (uint) | | |
| 函数描述 | 获得错误码 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_ret | uint | 返回错误码 |
| 接口原型 | function log(string \_str) constant public returns(uint \_ret) | | |
| 函数描述 | 日志输出1个字符串语句 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_str | string | 日志输出1个字符串语句 |
| 返回值 | \_ret | uint | 返回0 如：log(“Okay”) |
| 接口原型 | function log(string \_str, string \_str2) constant public returns(uint \_ret) | | |
| 函数描述 | 日志输出2个字符串语句 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_str | string | 第1个字符串语句 |
| \_str2 | string | 第2个字符串语句 |
| 返回值 | \_ret | uint | 返回0 如: log(“okay”,OwnerNamel”) |
| 接口原型 | function log(string \_str, string \_str2, string \_str3) constant public returns(uint \_ret) | | |
| 函数描述 | 日志输出3个字符串语句 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_str | string | 第1个字符串语句 |
| \_str2 | string | 第2个字符串语句 |
| \_str3 | string | 第3个字符串语句 |
| 返回值 | \_ret | uint | 返回0，输出成功 |
| 接口原型 | function log(string \_str, uint \_ui) constant public returns(uint \_ret) | | |
| 函数描述 | 日志输出1个字符串语句 + 1个uint型数据 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_str | string | 第1个字符串语句 |
| \_ui | uint | 输出的uint型数据 |
| 返回值 | \_ret | uint | 返回0，输出成功 |
| 接口原型 | function log(string \_str, int \_i) constant public returns(uint \_ret) | | |
| 函数描述 | 日志输出1个字符串语句 + 1个int型数据 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_str | string | 第1个字符串语句 |
| \_i | int | 输出的int型数据 |
| 返回值 | \_ret | uint | 返回0，输出成功 |
| 接口原型 | function log(string \_str, address \_addr) constant public returns(uint \_ret) | | |
| 函数描述 | 日志输出1个字符串语句 + 1个地址型数据 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_str | string | 第1个字符串语句 |
| \_addr | address | 输出的地址型数据 |
| 返回值 | \_ret | uint | 返回0，输出成功 |
| 接口原型 | function writedb(string \_name, string \_key, string \_value) public constant returns(uint \_ret) | | |
| 函数描述 | 将数据写入底层 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_name | string | 写入数据名称 |
| \_key | string | 数据key |
| \_value | string | 数据value |
| 返回值 | \_ret | uint | 返回0，写入成功 |

### RegisterManager.sol

此合约提供合约注册接口，所有业务合约均会在此合约注册。

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function register(string \_name, string \_version) | | |
| 函数描述 | 合约注册 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_name | string | 合约名称 |
| \_version | string | 合约版本 |
| 返回值 | 无 |  | 合约注册 |
| 接口原型 | function unRegister() | | |
| 函数描述 | 合约注销 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | 无 |  | 合约注销 |
| 接口原型 | function getContractAddress(string \_name,string \_version) constant returns (address \_address) | | |
| 函数描述 | 获取合约地址 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_name | string | 合约名称 |
| \_version | string | 合约版本 |
| 返回值 | \_address | address | 获取合约地址 |
| 接口原型 | function IfContractRegist(address \_contractAddr) constant returns(bool) | | |
| 函数描述 | 判断合约是否注册 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_contractAddr | address | 合约地址 |
| 返回值 | \_ret | bool | 判断合约是否注册 |
| 接口原型 | function findResNameByAddress(address \_addr) constant public returns(uint \_contractName) | | |
| 函数描述 | 返回合约名称 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_addr | address | 合约地址 |
| 返回值 | \_contractName | uint | 返回合约名称 |
| 接口原型 | function getRegisteredContract(uint \_pageNum, uint \_pageSize) constant public returns(string \_json) | | |
| 函数描述 | 返回已注册过的合约 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_pageNum | uint | 分页号 |
| \_pageSize | uint | 分页大小 |
| 返回值 | \_json | string | 返回已注册过的合约 |
| 接口原型 | function transferContract(string \_fromName, string \_fromVersion, string \_toName, string \_toVersion, string \_signString) public returns (uint \_errno) | | |
| 函数描述 | 转移合约 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_fromName | string | 被转移合约名称 |
| \_fromVersion | string | 被转移合约版本 |
| \_toName | string | 转移目标合约名称 |
| \_toVersion | string | 转移目标合约版本 |
| \_signString | string | 签名消息 |
| 返回值 | \_errno | uint | 返回错误码 |

### UserManager.sol

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function getUserState(address \_userAddr) constant public returns (uint \_state) | | |
| 函数描述 | 查找用户状态 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| 返回值 | \_state | uint | 返回用户状态 |
| 接口原型 | function getAccountState(string \_account) constant public returns (uint \_state) | | |
| 函数描述 | 查找账户状态 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_account | string | 用户账户 |
| 返回值 | \_state | uint | 返回账户状态 |
| 接口原型 | function pageByAccountStatus(uint \_accountStatus, uint \_pageNo, uint \_pageSize) public constant returns(string \_strjson) | | |
| 函数描述 | 根据账户状态分页查询用户数据 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_accountStatus | uint | 账户状态 |
| \_pageNo | uint | 分页号 |
| \_pageSize | uint | 分页大小 |
| 返回值 | \_strJson | string | 返回用户信息 |
| 接口原型 | function findByAddress(address \_userAddr) constant public returns(string \_ret) | | |
| 函数描述 | 根据用户地址查询用户 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| 返回值 | \_ret | String | 返回用户信息 |
| 接口原型 | function findByLoginName(string \_name) constant public returns(string \_strjson) | | |
| 函数描述 | 根据登录名查询用户 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_name | string | 用户登录名 |
| 返回值 | \_strjson | string | 返回用户信息 |
| 接口原型 | function findByAccount(string \_account) constant public returns(string \_strjson) | | |
| 函数描述 | 根据账户查询用户 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_account | string | 用户账户名 |
| 返回值 | \_strjson | string | 返回用户信息 |
| 接口原型 | function findByMobile(string \_mobile) constant public returns(string \_strjson) | | |
| 函数描述 | 根据手机查找用户 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_mobile | string | 用户手机号 |
| 返回值 | \_strjson | string | 返回用户信息 |
| 接口原型 | function findByEmail(string \_email) constant public returns(string \_strjson) | | |
| 函数描述 | 根据邮箱查找用户 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_email | string | 用户邮箱 |
| 返回值 | \_strjson | string | 返回用户信息 |
| 接口原型 | function log(string \_str, uint \_ui) constant public returns(uint \_ret) | | |
| 函数描述 | 日志输出类型字符串 +1 uint型数据 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_str | string | 输出的字符串 |
| \_ui | uint | 输出的uint型数据 |
| 返回值 | \_ret | uint | 返回0， 输出成功 |
| 接口原型 | function findByDepartmentId(string \_departmentId) constant public returns(string \_strjson) | | |
| 函数描述 | 根据组织Id查找用户 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_departmentId | string | 组织Id |
| 返回值 | \_strjson | string | 返回用户数据 |
| 接口原型 | function findByDepartmentIdTree(string \_departmentId, uint \_pageNum, uint \_pageSize) constant public returns(string \_strjson) | | |
| 函数描述 | 根据组织树查找用户 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_departmentId | string | 组织Id |
| \_pageNum | uint | 分页号 |
| \_pageSize | uint | 分页大小 |
| 返回值 | \_strjson | string | 根据组织树查找用户 |
| 接口原型 | function findByDepartmentIdTreeAndContion(uint \_status,string \_name,string \_departmentId, uint \_pageNum, uint \_pageSize) constant public returns(string \_strjson) | | |
| 函数描述 | 根据组织树以及查找条件查找用户 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_status | uint | 用户状态 |
| \_name | string | 用户名称 |
| \_departmentId | string | 组织Id |
| \_pageNum | uint | 分页号 |
| \_pageSize | uint | 分页大小 |
| 返回值 | \_strjson | string | 根据组织树以及查找条件查找用户 |
| 接口原型 | function findByRoleId(string \_roleId) constant public returns(string \_strjson) | | |
| 函数描述 | 根据角色Id查找用户 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_roleId | string | 角色Id |
| 返回值 | \_strjson | string | 根据角色Id查找用户 |
| 接口原型 | function getUserDepartmentId(address \_userAddr) constant returns(uint \_departId) | | |
| 函数描述 | 根据用户地址查找组织 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| 返回值 | \_departId | uint | 根据用户地址查找组织 |
| 接口原型 | function checkUserRole(address \_userAddr, string \_roleId) constant public returns(uint \_ret) | | |
| 函数描述 | 判断用户是否具备此角色 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| \_roleId | string | 角色Id |
| 返回值 | \_ret | uint | 判断用户是否具备此角色 |
| 接口原型 | function checkUserAction(address \_userAddr, string \_actionId) constant public returns (uint \_ret) | | |
| 函数描述 | 判断用户是否具备此权限 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| \_actionId | string | 权限Id |
| 返回值 | \_ret | uint | 判断用户是否具备此权限 |
| 接口原型 | function checkUserPrivilege(address \_userAddr, address \_contractAddr, string \_funcSha3) constant public returns (uint \_ret) | | |
| 函数描述 | 判断此用户是否可调用此函数 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| \_contractAddr | address | 合约地址 |
| \_funcSha3 | string | 此合约的函数 |
| 返回值 | \_ret | uint | 判断此用户是否可调用此函数 |
| 接口原型 | function userExists(address \_userAddr) constant public returns(uint \_ret) | | |
| 函数描述 | 判断此用户是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| 返回值 | \_ret | uint | 判断此用户是否存在 |
| 接口原型 | function insert(string \_userJson) public returns(uint) | | |
| 函数描述 | 新增用户 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userJson | string | 用户数据对象 |
| 返回值 | \_ret | uint | 新增用户 |
| 接口原型 | function update(string \_userJson) public returns(bool \_ret) | | |
| 函数描述 | 更新用户信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userJson | string | 用户数据对象 |
| 返回值 | \_ret | bool | 更新用户信息 |
| 接口原型 | function updateUserStatus(address \_userAddr, uint \_status) public returns(bool \_ret) | | |
| 函数描述 | 更新用户状态 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| \_status | uint | 用户状态 |
| 返回值 | \_ret | bool | 更新用户状态 |
| 接口原型 | function updateAccountStatus(address \_userAddr, uint \_status) public returns(bool \_ret) | | |
| 函数描述 | 更新账户状态 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| \_status | uint | 账户状态 |
| 返回值 | \_ret | bool | 更新账户状态 |
| 接口原型 | function updatePasswordStatus(address \_userAddr, uint \_status) public returns(bool \_ret) | | |
| 函数描述 | 更新口令状态 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| \_status | uint | 口令状态 |
| 返回值 | \_ret | bool | 更新口令状态 |
| 接口原型 | function addUserRole(address \_userAddr, string \_roleId) returns(uint) | | |
| 函数描述 | 用户添加角色 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| \_roleId | string | 角色Id |
| 返回值 | \_ret | uint | 用户添加角色 |
| 接口原型 | function deleteByAddress(address \_userAddr) public | | |
| 函数描述 | 删除用户 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| 返回值 | 无 |  | 删除用户 |
| 接口原型 | function login(string \_account) public returns(string \_json) | | |
| 函数描述 | 用户登录 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_account | string | 账户名 |
| 返回值 | \_json | string | 用户登录 |
| 接口原型 | function listAll() constant public returns(string \_userListJson) | | |
| 函数描述 | 查找所有用户列表 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_无 |  |  |
| 返回值 | \_userListjson | string | 查找所有用户列表 |
| 接口原型 | function getUserCount() constant public returns(uint \_count) | | |
| 函数描述 | 获取所有用户数量 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_count | uint | 获取所有用户数量 |
| 接口原型 | function getUserRoleId(address \_userAddr, uint \_index) constant returns (uint \_ret) | | |
| 函数描述 | 获取用户角色 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| \_index | uint | 角色索引号 |
| 返回值 | \_ret | uint | 获取用户角色 |
| 接口原型 | function getRevision() constant public returns(uint \_revision) | | |
| 函数描述 | 获取版本号 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_revision | uint | 获取版本号 |
| 接口原型 | function getUserCountByDepartmentId(string \_departmentId) constant public returns(uint \_count) | | |
| 函数描述 | 查找组织下的用户数目 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_departmentId | string | 组织Id |
| 返回值 | \_count | uint | 查找组织下的用户数目 |
| 接口原型 | function getUserCountByActionId(string \_actionId) constant public returns(uint \_count) | | |
| 函数描述 | 获取拥有指定权限的用户数量 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_actionId | string | 权限Id |
| 返回值 | \_count | uint | 获取拥有指定权限的用户数量 |
| 接口原型 | function roleUsed(string \_roleId) constant public returns (uint \_used) | | |
| 函数描述 | 检查role是否被使用 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_roleId | string | 角色Id |
| 返回值 | \_used | uint | 检查role是否被使用 |
| 接口原型 | function getUserCountMappingByRoleIds(string \_roleIds) constant public returns(string \_json) | | |
| 函数描述 | 根据角色ID获取用户数量 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_roleids | string | 角色列表 |
| 返回值 | \_json | string | 根据角色ID获取用户数量 |
| 接口原型 | function \_\_getDepartmentAdmin(string departmentId) constant internal returns(address admin) | | |
| 函数描述 | 获取组织管理员 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | departmentId | string | 组织Id |
| 返回值 | Admin | address | 获取组织管理员 |
| 接口原型 | function \_\_checkWritePermission(address \_addr, string \_departmentId) constant internal returns (uint \_ret) | | |
| 函数描述 | 判断该地址用户在该机构是否具有操作权限 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_addr | string | 用户地址 |
| \_departmentId | string | 组织Id |
| 返回值 | \_ret | uint | 0 有 1 无 |
| 接口原型 | function \_\_getDepartmentIdTree(string \_departmentId, string[] storage \_departmentIdTree) | | |
| 函数描述 | 获取组织Id树 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_departmentId | string | 组织Id |
| \_departmentIdTree | string[] | 组织树 |
| 返回值 | 无 |  | 获取组织Id树 |
| 接口原型 | function \_\_getRoleIdListByActionId(string \_actionId, string[] storage roleIdList) | | |
| 函数描述 | 根据权限Id获得相关的角色列表 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_actionId | string | 权限Id |
| roleIdList | String[] | 角色列表 |
| 返回值 | 无 |  | 根据权限Id获得相关的角色列表 |

### RoleManager.sol

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function insert(string \_json) public returns(uint) | | |
| 函数描述 | 新增角色 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 角色数据对象 |
| 返回值 | \_ret | uint | 新增角色 |
| 接口原型 | function update(string \_json) | | |
| 函数描述 | 更新角色信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 角色数据对象 |
| 返回值 | 无 |  | 更新角色信息 |
| 接口原型 | function listAll() constant public returns (string \_json) | | |
| 函数描述 | 查找所有角色 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_json | string | 查找所有角色 |
| 接口原型 | function findById(string \_id) constant public returns(string \_json) | | |
| 函数描述 | 根据角色Id查找相应角色 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_id | string | 角色Id |
| 返回值 | \_json | string | 根据角色Id查找相应角色 |
| 接口原型 | function findByName(string \_name) constant public returns(string \_json) | | |
| 函数描述 | 根据角色名称查找相应角色 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_name | string | 角色名称 |
| 返回值 | \_json | string | 根据角色名称查找相应角色 |
| 接口原型 | function checkRoleAction(string \_roleId, string \_actionId) constant public returns (uint \_ret) | | |
| 函数描述 | 判断角色是否具有相应权限 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_roleid | string | 角色Id |
| \_actionId | string | 权限Id |
| 返回值 | \_ret | uint | 判断角色是否具有相应权限 |
| 接口原型 | function roleExists(string \_roleId) constant public returns (uint \_ret) | | |
| 函数描述 | 判断角色是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_roleId | string | 角色Id |
| 返回值 | \_ret | uint | 判断角色是否存在 |
| 接口原型 | function actionUsed(string \_actionId) constant public returns (uint \_used) | | |
| 函数描述 | 判断权限是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_actionId | string | 权限Id |
| 返回值 | \_used | uint | 判断权限是否存在 |
| 接口原型 | function pageByName(string \_name, uint \_pageNum, uint \_pageSize) constant public returns (string \_json) | | |
| 函数描述 | 根据角色名称分页查询角色信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_name | string | 角色名称 |
| \_pageNum | uint | 分页号 |
| \_pageSize | uint | 分页大小 |
| 返回值 | \_json | string | 返回权限数据 |
| 接口原型 | function pageByNameAndModuleId(string \_moduleId,string \_name, uint \_pageNum, uint \_pageSize) constant public returns (string \_json) | | |
| 函数描述 | 根据角色名称和模块Id分页查询角色信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_moduleId | string | 模块Id |
| \_name | string | 角色名称 |
| \_pageNum | uint | 分页号 |
| \_pageSize | uint | 分页大小 |
| 返回值 | \_json | string | 返回权限数据 |
| 接口原型 | function checkRoleActionWithKey(string \_roleId, address \_resKey, string \_opKey) constant public returns (uint \_ret) | | |
| 函数描述 | 判断角色是否具有相应权限 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_roleid | string | 角色Id |
| \_resKey | string | 资源符 |
| \_opKey | string | 操作符 |
| 返回值 | \_ret | uint | 判断角色是否具有相应权限 |
| 接口原型 | function deleteById(string \_roleId) | | |
| 函数描述 | 删除角色 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_roleId | string | 角色Id |
| 返回值 | 无 |  | 删除角色 |
| 接口原型 | function getRoleIdByActionIdAndIndex(string \_actionId, uint \_index) constant public returns (uint \_roleId) | | |
| 函数描述 | 根据权限Id和索引号查找角色Id | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_actionId | string | 权限Id |
| \_index | uint | 索引号 |
| 返回值 | \_roleId | uint | 根据权限Id和索引号查找角色Id |
| 接口原型 | function \_\_roleUsed(string \_roleId) constant internal returns (uint \_ret) | | |
| 函数描述 | 判断角色是否使用 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_roleId | string | 角色Id |
| 返回值 | \_ret | uint | 判断角色是否使用 |
| 接口原型 | function \_\_actionExists(string \_actionId) constant internal returns (uint \_ret) | | |
| 函数描述 | 判断权限是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_actionId | string | 权限Id |
| 返回值 | \_ret | uint | 判断权限是否存在 |
| 接口原型 | function \_\_userExists(address \_userAddr) constant internal returns (uint \_ret) | | |
| 函数描述 | 判断用户是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| 返回值 | \_ret | uint | 判断用户是否存在 |
| 接口原型 | function \_\_getUserDepartmentId(address \_userAddr) constant internal returns (string \_ret) | | |
| 函数描述 | 根据用户地址查找用户组织Id | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| 返回值 | \_ret | string | 查找用户组织Id |
| 接口原型 | function \_\_getUserRoleIdList(address \_userAddr, string[] storage roleIdList) | | |
| 函数描述 | 获取用户角色列表 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| \_roleIdList | string[] | 角色列表 |
| 返回值 | 无 |  | 获取用户角色列表 |
| 接口原型 | function getActionIdListByRoleIdList(string[] storage roleIdList, string[] storage actionIdList) | | |
| 函数描述 | 根据角色列表活动权限列表 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | roleIdList | string[] | 角色列表 |
| actionIdList | string[] | 权限列表 |
| 返回值 | 无 |  | 根据角色列表活动权限列表 |
| 接口原型 | function getRoleModuleId(string \_roleId) constant public returns (uint \_ret) | | |
| 函数描述 | 根据角色Id活动模块Id | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_roleId | string | 角色Id |
| 返回值 | \_ret | uint | 根据角色Id活动模块Id |

### DepartmentManager.sol

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function insert(string \_json) public returns(uint) | | |
| 函数描述 | 新增组织 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 组织数据对象 |
| 返回值 | \_ret | uint | 新增组织 |
| 接口原型 | function update(string \_json) public returns(uint) | | |
| 函数描述 | 更新组织信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 组织数据对象 |
| 返回值 | \_ret | uint | 更新组织信息 |
| 接口原型 | function setDepartmentStatus(string \_departmentId, uint \_status) public returns(uint) | | |
| 函数描述 | 设置组织状态 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_departmentId | string | 组织Id |
| \_status | uint | 组织状态 |
| 返回值 | \_ret | uint | 设置组织状态 |
| 接口原型 | function setAdmin(string \_departmentId, address \_adminAddr) public returns(uint) | | |
| 函数描述 | 设置组织管理员 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_departmentId | string | 组织Id |
| \_adminAddr | address | 管理员地址 |
| 返回值 | \_ret | uint | 设置组织管理员 |
| 接口原型 | function eraseAdminByAddress(address \_userAddr) | | |
| 函数描述 | 删除组织管理员 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| 返回值 | 无 |  | 删除组织管理员 |
| 接口原型 | function departmentExistsByCN(string \_commonName) constant public returns (uint \_ret) | | |
| 函数描述 | 根据组织CN判断组织是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_commonName | string | 组织CN |
| 返回值 | \_ret | uint | 根据组织CN判断组织是否存在 |
| 接口原型 | function departmentExists(string \_departmentId) constant public returns (uint \_exists) | | |
| 函数描述 | 根据组织Id判断组织是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_departmentId | string | 组织Id |
| 返回值 | \_exists | uint | 根据组织Id判断组织是否存在 |
| 接口原型 | function departmentEmpty(string \_departmentId) constant public returns (bool \_empty) | | |
| 函数描述 | 判断组织是否空 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_departmentId | string | 组织Id |
| 返回值 | \_empty | bool | 判断组织是否空 |
| 接口原型 | function deleteById(string \_departmentId) | | |
| 函数描述 | 根据组织Id删除组织 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_departmentId | string | 组织Id |
| 返回值 | 无 |  | 根据组织Id删除组织 |
| 接口原型 | function listAll() constant public returns (string \_json) | | |
| 函数描述 | 查找所有组织 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_json | string | 查找所有组织 |
| 接口原型 | function findById(string \_id) constant public returns(string \_json) | | |
| 函数描述 | 根据组织Id查询组织信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_id | string | 组织Id |
| 返回值 | \_json | string | 根据组织Id查询组织信息 |
| 接口原型 | function findByName(string \_name) constant public returns(string \_json) | | |
| 函数描述 | 根据组织名称查询组织信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_name | string | 组织名称 |
| 返回值 | \_json | string | 根据组织名称查询组织信息 |
| 接口原型 | function findByParentId(string \_parentId) constant public returns(string \_json) | | |
| 函数描述 | 根据父组织Id查询组织信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_parentId | string | 父组织Id |
| 返回值 | \_json | string | 根据父组织Id查询组织信息 |
| 接口原型 | function getAdmin(string \_departmentId) constant public returns(uint \_admin) | | |
| 函数描述 | 查询组织管理员 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | departmentId | string | 组织Id |
| 返回值 | \_admin | uint | 查询组织管理员 |
| 接口原型 | function getRevision() constant public returns(uint \_revision) | | |
| 函数描述 | 查询版本号 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_revision | uint | 查询版本号 |
| 接口原型 | function pageByName(string \_name, uint \_pageNum, uint \_pageSize) constant public returns(string \_json) | | |
| 函数描述 | 根据组织名称分页查询组织信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_name | string | 组织名称 |
| \_pageNum | uint | 分页号 |
| \_pageSize | uint | 分页大小 |
| 返回值 | \_json | string | 返回组织数据 |
| 接口原型 | function pageByNameAndStatus(string \_parentId,uint \_status, string \_name, uint \_pageNum, uint \_pageSize) constant public returns(string \_json) | | |
| 函数描述 | 根据组织名称和状态分页查询组织信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_parentId | string | 父组织Id |
|  | \_status | uint | 组织状态 |
|  | \_name | string | 组织名称 |
|  | \_pageNum | uint | 分页号 |
|  | \_pageSize | uint | 分页大小 |
| 返回值 | \_json | string | 返回组织数据 |
| 接口原型 | function getChildIdByIndex(string \_departmentId, uint \_index) constant public returns (uint \_childDepartmentId) | | |
| 函数描述 | 根据组织Id和索引号查找子组织Id | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_departmentId | string | 组织Id |
| \_index | uint | 索引号 |
| 返回值 | \_childDepartmentId | uint | 根据组织Id和索引号查找子组织Id |
| 接口原型 | function checkWritePermission(address \_addr, string \_departmentId) constant public returns (uint \_ret) | | |
| 函数描述 | 判断用户是否有操作权限 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_addr | address | 用户地址 |
| \_departmentId | String | 组织Id |
| 返回值 | \_ret | uint | 判断用户是否有操作权限 |
| 接口原型 | function getIndexById(string \_id) constant private returns (uint) | | |
| 函数描述 | 获得此组织的索引号 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_id | string | 组织Id |
| 返回值 | \_ret | uint | 获得此组织的索引号 |
| 接口原型 | function getIndexByCommonName(string \_commonName) constant private returns (uint) | | |
| 函数描述 | 根据组织CN获得索引号 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_commonName | string | 组织CN |
| 返回值 | \_ret | uint | 根据组织CN获得索引号 |
| 接口原型 | function \_\_roleExists(string \_roleId) constant internal returns (uint \_ret) | | |
| 函数描述 | 判断角色是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_roleId | string | 角色Id |
| 返回值 | \_ret | uint | 判断角色是否存在 |
| 接口原型 | function \_\_userExists(address \_userAddr) constant internal returns (uint \_ret) | | |
| 函数描述 | 判断用户是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| 返回值 | \_ret | uint | 判断用户是否存在 |
| 接口原型 | function \_\_getUserDepartmentId(address \_userAddr) constant internal returns (string \_ret) | | |
| 函数描述 | 根据用户地址获得用户组织Id | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| 返回值 | \_ret | string | 获得用户组织Id |
| 接口原型 | function \_\_getUserCountByDepartmentId(string \_departmentId) constant internal returns(uint \_count) | | |
| 函数描述 | 根据组织id获得组织下的用户数量 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_departmentId | string | 组织Id |
| 返回值 | \_count | uint | 获得组织下的用户数量 |

### ActionManager.sol

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function getActionListByModuleId(string \_moduleId) constant public returns(string \_json) | | |
| 函数描述 | 根据模块Id获得模块下的权限列表 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_moduleId | string | 模块Id |
| 返回值 | \_json | string | 获得模块下的权限列表 |
| 接口原型 | function getActionListByContractId(string \_contractId) constant public returns(string \_json) | | |
| 函数描述 | 根据合约id获得合约下的权限列表 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_contractId | string | 合约Id |
| 返回值 | \_json | string | 获得合约下的权限列表 |
| 接口原型 | function actionExists(string \_actionId) constant public returns(uint \_ret) | | |
| 函数描述 | 判断权限是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_actionId | string | 权限Id |
| 返回值 | \_ret | uint | 判断权限是否存在 |
| 接口原型 | function findByKey(string \_resKey, string \_opKey) constant public returns(string \_actionJson) | | |
| 函数描述 | 返回权限信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_resKey | string | 资源符（合约名称） |
| \_opkey | string | 操作符（函数名） |
| 返回值 | \_actionJson | string | 返回权限信息 |
| 接口原型 | function findById(string \_actionId) constant public returns(string \_actionJson) | | |
| 函数描述 | 根据权限Id查找权限信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_actionId | string | 权限id |
| 返回值 | \_actionJson | string | 根据权限Id查找权限信息 |
| 接口原型 | function listAll() constant public returns(string \_actionListJson) | | |
| 函数描述 | 查找所有权限列表 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_actionListJson | string | 查找所有权限列表 |
| 接口原型 | function listContractActions(string \_contractName) constant public returns(string \_actionListJson) | | |
| 函数描述 | 查询合约下的所有权限 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_contractName | string | 合约名称 |
| 返回值 | \_actionListJson | string | 查询合约下的所有权限 |
| 接口原型 | function checkActionWithKey(string \_actionId, address \_contractAddr, string \_opSha3Key) constant public returns(uint \_ret) | | |
| 函数描述 | 判断函数是否是此合约下的权限 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_actionId | string | 权限Id |
| \_contractAddr | address | 合约地址 |
| \_opSha3Key | string | 合约函数 |
| 返回值 | \_ret | uint | 判断函数是否是此合约下的权限 |
| 接口原型 | function insert(string \_actionJson) public returns(bool \_ret) | | |
| 函数描述 | 新增权限 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_actionJson | string | 权限数据对象 |
| 返回值 | \_ret | bool | 新增权限 |
| 接口原型 | function deleteById(string \_actionId) | | |
| 函数描述 | 根据权限Id删除权限信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_actionId | string | 权限id |
| 返回值 | 无 |  | 根据权限Id删除权限信息 |
| 接口原型 | function getCount() constant returns(uint \_count) | | |
| 函数描述 | 查询权限数量 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_count | uint | 查询权限数量 |

### NodeInfoManager.sol

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function insert(string \_json) public returns(uint) | | |
| 函数描述 | 新增节点 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 节点json数据对象 |
| 返回值 | \_ret | uint | 返回相应错误码 |
| 接口原型 | function update(string \_json) public returns(uint) | | |
| 函数描述 | 根据合约id获得合约下的权限列表 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 节点json数据对象 |
| 返回值 | \_ret | uint | 返回相应错误码 |
| 接口原型 | function getEnodeList() constant public returns(string \_json) | | |
| 函数描述 | 查找节点IP信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_json | string | 返回节点IP的json字符串 |
| 接口原型 | function ActivateEnode(string \_pubkey) public | | |
| 函数描述 | 激活节点 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_pubkey | string | 节点公钥 |
| 返回值 | 无 |  |  |
| 接口原型 | function isInWhiteList(string \_commonName, string \_ip) constant public returns (string \_json) | | |
| 函数描述 | 判断节点是否在白名单 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_commonName | string | 节点证书CN |
| \_ip | string | 节点IP |
| 返回值 | \_json | string | 是：”true” 否： “false” |
| 接口原型 | function getNodeAdmin(string \_nodeId) constant public returns(uint \_admin) | | |
| 函数描述 | 查找节点管理员 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_nodeId | string | 节点ID |
| 返回值 | \_admin | uint | 节点管理员地址 |
| 接口原型 | function setAdmin(string \_nodeId, address \_adminAddr) public returns(uint) | | |
| 函数描述 | 设置节点管理员 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_nodeId | string | 节点id |
| \_adminAddr | string | 管理员地址 |
| 返回值 | \_ret | uint | 返回相应错误码 |
| 接口原型 | function eraseAdminByAdd(address \_userAddr) public returns(uint) | | |
| 函数描述 | 删除管理员 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_userAddr | address | 用户地址 |
| 返回值 | \_ret | uint | 返回相应错误码 |
| 接口原型 | function nodeInfoExists(string \_nodeId) constant public returns (uint \_exists) | | |
| 函数描述 | 节点是否存在 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_nodeId | string | 节点id |
| 返回值 | \_exists | uint | 0 存在 1 不存在 |
| 接口原型 | function deleteById(string \_nodeId) | | |
| 函数描述 | 根据节点Id删除节点信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_nodeId | string | 节点id |
| 返回值 | 无 |  |  |
| 接口原型 | function getRevision() constant public returns (uint \_ret) | | |
| 函数描述 | 查找版本号 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_ret | uint | 返回版本号 |
| 接口原型 | function IPUsed(string \_ip) constant public returns (uint \_used) | | |
| 函数描述 | 检查IP是否已使用 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_ip | String | ip |
| 返回值 | \_userd | uint | 1 已使用 0 未使用 |
| 接口原型 | function listAll() constant public returns (string \_json) | | |
| 函数描述 | 查找所有节点信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_json | string | 返回所有节点信息json字符串 |
| 接口原型 | function findById(string \_id) constant public returns(string \_json) | | |
| 函数描述 | 根据节点id查询节点信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_id | string | 节点id |
| 返回值 | \_json | string | 返回节点信息 |
| 接口原型 | function findByName(string \_name) constant public returns(string \_json) | | |
| 函数描述 | 根据节点名称查询节点信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_name | string | 节点名称 |
| 返回值 | \_json | string | 返回节点信息 |
| 接口原型 | function findByDepartmentId(string \_departmentId) constant public returns(string \_json) | | |
| 函数描述 | 根据节点组织id查询节点信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_departmentId | string | 组织id |
| 返回值 | \_json | string | 返回节点信息 |
| 接口原型 | function findByNodeAdmin(address \_nodeAdmin) constant public returns(string \_json) | | |
| 函数描述 | 根据节点管理员查询节点信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_nodeAdmin | address | 节点管理员地址 |
| 返回值 | \_json | string | 返回节点信息 |
| 接口原型 | function findByPubkey(string \_pubkey) constant public returns(string \_json) | | |
| 函数描述 | 根据节点公钥查询节点信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_pubkey | string | 节点公钥 |
| 返回值 | \_json | string | 返回节点信息 |
| 接口原型 | function checkWritePermission(address \_addr, string \_nodeInfoId) constant public returns (uint \_ret) | | |
| 函数描述 | 判断用户是否由写入权限 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_addr | address | 用户地址 |
| \_nodeInfoId | string | 节点Id |
| 返回值 | \_ret | uint | 0 有权限 1 无权限 |

### 4.2.8.NodeApplyManager.sol

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function insert(string \_json) public returns(uint) | | |
| 函数描述 | 新增节点申请 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 节点申请信息 |
| 返回值 | \_ret | uint | 返回相应错误码 |
| 接口原型 | function update(string \_json) public returns(uint) | | |
| 函数描述 | 更新节点申请信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 节点申请信息 |
| 返回值 | \_ret | uint | 返回相应错误码 |
| 接口原型 | function nodeApplyExists(string \_nodeApplyId) constant public returns(uint \_exists) | | |
| 函数描述 | 判断节点是否已申请 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_nodeApplyId | string | 节点申请Id |
| 返回值 | \_exists | uint | 0 已申请 1 未申请 |
| 接口原型 | function auditing(string \_json) public returns(uint) | | |
| 函数描述 | 节点申请审核 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
|  | \_json | string | 节点申请信息 |
| 返回值 | \_ret | uint | 返回相应错误码 |
| 接口原型 | function pageByNameAndStatus(uint \_status, string \_deptName, uint \_pageNum, uint \_pageSize) constant public returns(string \_json) | | |
| 函数描述 | 根据名称和状态分页查询节点申请信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_status | uint | 节点申请状态 |
| \_deptName | string | 组织名称 |
| \_pageNum | uint | 页码 |
| \_pageSize | uint | 分页大小 |
| 返回值 | \_json | string | 返回节点申请信息 |
| 接口原型 | function listAll() constant public returns(string \_json) | | |
| 函数描述 | 查找所有节点申请列表 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | 无 |  |  |
| 返回值 | \_json | string | 返回所有节点申请列表 |
| 接口原型 | function findByState(uint \_state) constant public returns(string \_strjson) | | |
| 函数描述 | 根据节点申请状态查找节点申请信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_state | uint | 申请状态 |
| 返回值 | \_strjson | string | 返回节点申请信息 |
| 接口原型 | function findByApplyId(string \_nodeApplyId) constant public returns (string \_json) | | |
| 函数描述 | 根据节点申请id查找申请信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_nodeApplyId | string | 申请Id |
| 返回值 | \_json | string | 返回节点申请信息 |
| 接口原型 | function deleteById(string \_nodeApplyId) public returns(bool) | | |
| 函数描述 | 删除节点申请信息 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_nodeApplyId | string | 申请id |
| 返回值 | \_ret | bool | true 成功 false 失败 |

### BaseModule.sol

|  |  |  |  |
| --- | --- | --- | --- |
| 接口原型 | function addModule(string \_json) internal returns(uint \_ret) | | |
| 函数描述 | 新增模块 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 模块对象 |
| 返回值 | \_ret | uint | 返回的错误码 |
| 接口原型 | function addContract(string \_json) internal returns(uint \_ret) | | |
| 函数描述 | 新增合约 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 合约对象 |
| 返回值 | \_ret | uint | 返回的错误码 |
| 接口原型 | function addMenu(string \_json) internal returns(uint \_ret) | | |
| 函数描述 | 新增菜单 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 菜单对象 |
| 返回值 | \_ret | uint | 返回的错误码 |
| 接口原型 | function addAction(string \_json) internal returns(uint \_ret) | | |
| 函数描述 | 新增权限 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 权限对象 |
| 返回值 | \_ret | uint | 返回的错误码 |
| 接口原型 | function addRole(string \_json) internal returns(uint \_ret) | | |
| 函数描述 | 新增角色 | | |
|  | 参数名称 | 参数类型 | 取值说明 |
| 输入参数 | \_json | string | 角色对象 |
| 返回值 | \_ret | uint | 返回的错误码 |

## 系统角色与权限说明

系统内置四种角色以及每个角色对应的权限，分别为：节点管理员、链管理员、系统管理员、权限管理员，每个角色以及其对应的权限说明如下：

|  |  |  |
| --- | --- | --- |
| 角色名称 | **节点管理员** | |
| 角色权限 | 权限描述 | 四大合约权限以及节点信息管理权限 |
| 所有权限 | UserManager, DepartmentManager, ActionManager, RoleManager, NodeInfoManager |
| 角色名称 | **链管理员** | |
| 角色权限 | 权限描述 | 四大合约权限以及节点申请权限 |
| 所有权限 | UserManager, DepartmentManager, ActionManager, RoleManager, NodeApplyManager |
| 角色名称 | **系统管理员** | |
| 角色权限 | 权限描述 | 四大合约权限 |
| 所有权限 | UserManager, DepartmentManager, ActionManager, RoleManager |
| 角色名称 | **权限管理员** | |
| 角色权限 | 权限描述 | 四大合约权限控制 |
| 所有权限 | UserManager, DepartmentManager, ActionManager, RoleManager |

# 5.合约业务开发流程介绍

使用智能合约开发业务，需按照以下5个步骤进行：

1）完成合约基础数据的处理；

2）实现注册合约；

3）定义业务数据的结构，并实现对业务数据的处理；

4）根据接口文档实现业务合约的业务逻辑；

5）模块化管理。

下面将详细介绍这5个步骤。

## 5.1完成合约基础数据的处理

合约基础数据为int, string类型，因此首先需要实现对这些基础数据的各种处理，如：

### **5.1.1 string型数据截取：**

function substr(string \_self, uint \_start, uint \_len) internal returns (string \_ret) {

if (\_len > bytes(\_self).length-\_start) {

\_len = bytes(\_self).length-\_start;

}

if (\_len <= 0) {

\_ret = "";

return;

}

\_ret = new string(\_len);

uint selfptr;

uint retptr;

assembly {

selfptr := add(\_self, 0x20)

retptr := add(\_ret, 0x20)

}

memcpy(retptr, selfptr+\_start, \_len);

}

### **5.1.2 int型数据转string型：**

function toString(int \_self) internal returns (string \_ret) {

if (\_self == 0) {

return "0";

}

uint ui = uint(\_self);

bool positive = true;

uint8 len = 0;

if (\_self < 0) {

ui = uint(-\_self);

positive = false;

len++;

}

uint tmp = ui;

while (tmp > 0) {

tmp /= 10;

len++;

}

\_ret = new string(len);

if (!positive) {

bytes(\_ret)[0] = '-';

}

uint8 i = len-1;

while (ui > 0) {

bytes(\_ret)[i--] = byte(ui%10+0x30);

ui /= 10;

}

}

总的来说这一步，就是完成基础数据的处理来满足所有需求。

## 5.2实现注册合约

所有的业务合约都需在注册合约中注册，注册合约需实现合约注册、注销、获取合约地址等接口，从而所有注册过的发布入链合约，均能通过注册合约找到其在链上的地址，注册合约接口如：

注册信息由Register结构体构成：

struct Register {

string name; //合约名称

string version; //合约版本号

address addr; //合约地址

address origin; //coinbase地址

}

注册接口示例代码：

function register(string \_name, string \_version) {

address sendAddr = msg.sender;

Register register = registerMap[sendAddr];

register.name = \_name;

register.version = \_version;

register.addr = sendAddr;

register.origin= tx.origin;

string memory key = \_name.concat(\_version);

keyMap[key] = sendAddr;

contractAddrList.push(sendAddr);

}

如发布组织合约DepartmentManager，需在其构造函数内实现：

function DepartmentManager() {

register("DepartmentManager", "0.0.1.0");

}

这里register函数是通过先在待发布合约中实例化注册合约，然后再调用注册合约的register函数。

## 5.3定义业务数据结构

要完成所有业务合约，首先得先定义业务的数据结构，对应于合约的就是业务合约相应的Lib库。现有权限模型包含了组织、角色、用户、权限四大合约，以及管理节点信息的合约和处理节点审核信息的合约。

### **5.3.1组织合约，组织的数据结构：**

struct Department {

string id; //组织Id

string name; //组织名称

uint departmentLevel; //组织层级：0 链本身1 链下的一级组织2 一级组织下的二级组织

string parentId; //父级组织Id

string description; //组织描述

uint creTime; //组织创建时间

uint updTime; //组织更新时间

string commonName; //组织证书CN

string stateName; //组织省份

string countryName; //组织国家

address admin; //组织管理员

address creator; //组织创建者

bool deleted; //组织是否删除：false 未删除 true 删除

string orgaShortName; //组织简称

uint status; //组织状态：0 禁用 1 激活

string groupPubkey; //群公钥

}

### **5.3.2角色合约，角色的数据结构：**

struct Role {

string id; //角色Id

string name; //角色名称

uint status; //角色状态

string moduleId; //角色所属模块Id

string contractId; //角色所属合约Id

string description; //角色描述

uint creTime; //角色创建时间

uint updTime; //角色更新时间

bool deleted; //角色是否删除：false 未删除 true 删除

string[] actionIdList; //角色的权限列表

address creator; //角色创建者

}

### **5.3.3用户合约，用户的数据结构：**

struct User {

address userAddr; //用户地址

string name; //用户名称

string account; //用户账户

string email; //用户邮箱

string mobile; //用户手机

string departmentId; //用户所属组织ID

uint accountStatus; //用户账户状态：0 解锁 1 锁定

uint passwordStatus; //用户口令状态： //待定？？？？

uint deleteStatus; //用户删除状态：0 删除 1 未删除 【delete】 这个？？？？

string uuid; //用户的uuid

string publicKey; //用户公钥

string cipherGroupKey; //用自己私钥加密群私钥后的密文

uint createTime; //用户创建时间

uint updateTime; //用户更新时间

uint loginTime; //用户登录时间

string[] roleIdList; //用户角色列表

UserState state; //标记用户数据是否有效

uint status; //是否禁用用户：0 禁用 1 激活

}

### **5.3.4权限合约，权限的数据结构：**

struct Action {

string id; //权限ID

string name; //权限名称

string contractId; //权限所属合约Id

string moduleId; //权限所属模块Id

uint level; //权限层级：1 1级菜单 2 2级菜单 3 合约的函数接口

uint Type; //权限类型：0按钮 1 菜单 2 合约函数

uint enable; //是否启用：0 未启用 1 启用

string parentId; //权限父Id

string url; //权限资源定位器

string description; //权限描述

string resKey; //权限所属合约的合约名称，若权限是指菜单，则为菜单名称

string opKey; //权限所指的合约函数，若权限是指菜单，则用阿拉伯数字

string version; //版本号

uint createTime; //权限创建时间

uint updateTime; //权限更新时间

ActionState state; //权限状态：0 无效 1 有效

address creator; //权限创建者

}

业务数据对象均以Json格式存储，因此在业务合约对应的Lib库合约中实现对业务数据的Json解析。

## 5.4根据接口文档实现业务逻辑

业务合约首先定义好接口，其次业务合约需定义好每个合约的错误码，以供开发人员进行调试，满足业务各种异常情况处理，举用户合约的例子来说，用户合约的错误码定义为：

enum UserError {

NO\_ERROR, //0 无误

BAD\_PARAMETER, //1 参数不对

NAME\_EMPTY, //2 用户名字为空

DEPT\_NOT\_EXISTS, //3 组织Id不存在

ROLE\_ID\_INVALID, //4 角色ID无效

USER\_NOT\_EXISTS, //5 用户数据不存在

ROLE\_ID\_ALREADY\_EXISTS, //6 角色已存在

ADDRESS\_ALREADY\_EXISTS, //7 地址已存在

ACCOUNT\_ALREADY\_EXISTS, //8 账号已存在

ACCOUNT\_CANNOT\_UPDATE, //9 账户不可被更新

USER\_LOGIN\_FAILED, //10用户登录失败

DEPT\_CANNOT\_UPDATE, //11用户组织信息不可更新

NO\_PERMISSION //12无权限

}

通常每个业务合约固定一个错误码偏移量，如用户合约参数不对，返回错误码error = 15000 + UserError.BAD\_PARAMETER. 用户合约需实现用户的增加，删除，查询，用户信息更新等功能，新增用户接口传参为string \_userJson, 即用户数据，如

{

"userAddr": "0x336d8f5f0c24bca870bf2b4f09741a896e757f27",

"name": "name0001",

"account": "applyUser0001",

"email": "juzhen@juzix.io",

"mobile": "13400001111",

"departmentId": "department0001",

"passwordStatus": 0,

"accountStatus": 0,

"deleteStatus": 0,

"uuid": "uuid0001",

"publicKey": "publicKey0001",

"cipherGroupKey": "cipherGroupKey0001",

"updateTime": 0,

"createTime": 0,

"loginTime": 0,

"roleIdList": ["role100002"],

"state": 0,

"status": 0

}

将其转换成string型作为传参，新增用户成功的话，返回0；反之，新增失败则根据返回的错误码，可以找到对应的缘由。

## 5.5模块化管理

模块化管理，有助于进行合约管理，方便地实现合约的可插拔性。

合约的数据结构为：

struct Contract{

string moduleId; //所属模块ID

string contractId; //合约ID

string contractName; //合约名称

string contractVersion; //合约版本号

bool deleted; //合约是否删除：false删除 true 未删除

uint enable; //合约是否启用：0 未启用 1 启用

string description; //合约描述

uint createTime; //合约创建时间

uint updateTime; //合约更新时间

address creator; //合约创建者

address contractAddr; //合约地址

}

每个业务合约均带有一个模块ID，表示归属于此模块，目前所有业务合约均挂在系统模块下。模块下同时挂载了所有的权限，这里权限指的是所有合约的逻辑处理；也挂载了所有菜单，这里菜单指的是权限的功能集合；同时模块下也挂载了开发时设定的角色，即内置角色，每个角色分配好其所具有的权限，给每个用户分配权限时，只需给其分配相应的角色，用户就可获得此角色下的所有权限。

现在我们所有业务合约均归属于系统模块SystemModule，因此在系统模块合约SystemModuleManager中，将所有业务合约、菜单、权限、角色加入到系统模块中，在此合约的构造函数中实现：

function SystemModuleManager(){

register("SystemModuleManager","0.0.1.0");

initContractData(); //添加合约数据

initMenuData(); //添加菜单数据

initActionData(); //添加权限数据

initRoleData(); //添加角色数据

}

这样就将所有需求功能内置，在做实现应用时直接调取相应接口就可实现所需功能。