**lua合约开发指南**

#### 目录

[1 常量 3](#_Toc464039783)

[1.1 --脚本应用账户操作类型定义 3](#_Toc464039784)

[1.2 --系统账户操作定义 3](#_Toc464039785)

[1.3 --日志类型 4](#_Toc464039786)

[1.4 --账户类型 4](#_Toc464039787)

[2 接口函数 5](#_Toc464039788)

[2.1 LogPrint 5](#_Toc464039789)

[2.2 Sha256 6](#_Toc464039790)

[2.3 Des 6](#_Toc464039791)

[2.4 VerifySignature 7](#_Toc464039792)

[2.5 GetTxContracts 8](#_Toc464039793)

[2.6 GetTxAccounts 9](#_Toc464039794)

[2.7 GetAccountPublickey 10](#_Toc464039795)

[2.8 QueryAccountBalance 10](#_Toc464039796)

[2.9 GetTxConFirmHeight 11](#_Toc464039797)

[2.10 GetBlockHash(height) 12](#_Toc464039798)

[2.11 GetCurRunEnvHeight 12](#_Toc464039799)

[2.12 WriteData 13](#_Toc464039800)

[2.13 DeleteData 14](#_Toc464039801)

[2.14 ReadData 14](#_Toc464039802)

[2.15 ModifyData 15](#_Toc464039803)

[2.16 GetCurTxHash 16](#_Toc464039804)

[2.17 WriteOutput 17](#_Toc464039805)

[2.18 GetScriptData 18](#_Toc464039806)

[2.19 GetScriptID 19](#_Toc464039807)

[2.20 GetCurTxAccount 19](#_Toc464039808)

[2.21 GetCurTxPayAmount 20](#_Toc464039809)

[2.22 GetUserAppAccValue 21](#_Toc464039810)

[2.23 GetUserAppAccFoudWithTag 21](#_Toc464039811)

[2.24 WriteOutAppOperate 22](#_Toc464039812)

[2.25 GetBase58Addr 23](#_Toc464039813)

[2.26 ByteToInteger 24](#_Toc464039814)

[2.27 IntegerToByte4 25](#_Toc464039815)

[2.28 IntegerToByte8 25](#_Toc464039816)

[2.29 TransferContactAsset 26](#_Toc464039817)

[2.30 TransferSomeAsset 27](#_Toc464039818)

[3 注意事项 28](#_Toc464039819)

[3.1由于安全性考虑，合约中只能使用lua中table，math，string库，不能使用io操作等。 28](#_Toc464039820)

[3.2脚本现在只能写在一个文件中，不支持多文件。 28](#_Toc464039821)

[3.3脚本大小不能超过64k。 28](#_Toc464039822)

[3.4合约内容通过全局变量“contract”传给脚本，脚本里面可以拿来就用 28](#_Toc464039823)

[3.5账户平衡检测开关通过在lua脚本中的全局变量“gCheckAccount”控制，默认关闭此开关。 28](#_Toc464039824)

[3.6合约内容不能超过4096字节。 28](#_Toc464039825)

[3.7脚本中如果有异常，或逻辑不对，可以通过assert，error退出执行，lua解释器会捕获到该异常。 28](#_Toc464039826)

[3.8脚本中必须以mylib = require "mylib"开头，通过mylib调用里面的API。 28](#_Toc464039827)

[4 示例 （充值和提现操作） 28](#_Toc464039828)

[5 脚本调试 38](#_Toc464039829)

[5.1 启动dacrs程序 39](#_Toc464039830)

[5.2 导入私钥 39](#_Toc464039831)

[5.3 生成几个测试地址，并充值激活 40](#_Toc464039832)

[5.4 注册应用脚本 42](#_Toc464039833)

[5.5 充值操作 43](#_Toc464039834)

[5.6 提现操作 48](#_Toc464039835)

[5.5.1查看dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6当前余额 ： 49](#_Toc464039836)

[5.5.2提现 49](#_Toc464039837)

[5.5.3 再次查看dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6当前余额 ，发现已经提现了5个币。 49](#_Toc464039838)

[5.7 脚本出错处理 50](#_Toc464039839)

[6 部署 50](#_Toc464039840)

开发规范请参考《lua脚本规范.txt》

RPC接口请参考《RPCCommandV0.1 beta.docx》

原理请参考《DACRS系统智能合约LUA应用开发原理.docx》

# 常量

## 1.1 --脚本应用账户操作类型定义

APP\_OPERATOR\_TYPE =

{

ENUM\_ADD\_FREE\_OP = 1, --自由账户加

ENUM\_SUB\_FREE\_OP = 2, --自由账户减

ENUM\_ADD\_FREEZED\_OP = 3, --冻结账户加

ENUM\_SUB\_FREEZED\_OP = 4 --冻结账户减

}

## 1.2 --系统账户操作定义

OPER\_TYPE =

{

ENUM\_ADD\_FREE = 1, --系统账户加

ENUM\_MINUS\_FREE = 2 --系统账户减

}

## 1.3 --日志类型

LOG\_TYPE =

{

ENUM\_STRING = 0, --字符串类型

ENUM\_NUMBER = 1 --数字类型

}

## 1.4 --账户类型

ADDR\_TYPE =

{

ENUM\_REGID = 1, -- REG\_ID

ENUM\_BASE58 = 2 -- BASE58 ADDR

}

# 接口函数

## 2.1 LogPrint

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.LogPrint(LogTable) | |
| 功能 | | 保存日志信息 | |
| 参数 | 输入 | 待保存的内容 |  |
| 输出 |  | |
| 常量 | | --日志输出类型  LOG\_TYPE =  {  ENUM\_STRING = 0, --字符串类型  ENUM\_NUMBER = 1 --数字类型  }; | |
| 结构体 | |  | |
| 返回 | |  |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ LogPrint ()  local LogTable = {  key = LOG\_TYPE.ENUM\_STRING, --日志类型  length = 0, --value数据流的总长  value = nil -- 字符串或数字流  }  --保存字符串  LogTable.length = 9  LogTable.value = "lua start"  mylib.LogPrint(LogTable)  --保存数据流  LogTable.key = LOG\_TYPE.ENUM\_NUMBER  LogTable.length = 20  local i  for i = 1,20 do  LogTable.value [i] = i  end  mylib.LogPrint(LogTable)  end | |

## 2.2 Sha256

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.Sha256(string) | |
| 功能 | | Sha256 hash加密算法 | |
| 参数 | 输入 | 待加密的内容 |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 加密输出，成功返回一个table，否则nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_Sha256 ()  local orgContent = "123"  local content = {mylib.Sha256(orgContent)}  LogPrint(LOG\_TYPE.ENUM\_NUMBER,#content,content)  end | |

## 2.3 Des

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.Des(desTbl) | |
| 功能 | | Des加密算法 | |
| 参数 | 输入 | 待加密的内容 |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | | local desTbl =  {  dataLen = 0, --加密数据长度  data = {}, --加密数据  keyLen = 0, --私钥长度  key = {}, --私钥数据  flag = 1 --1加密 0解密  } | |
| 返回 | | 加密输出，成功返回一个table，否则nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_Des ()  local desTbl =  {  dataLen = 8,  data = {},  keyLen = 8,  key = {},  flag = 1  }  desTbl.data = {0xad, 0xdd, 0x1e, 0x1b, 0xeb, 0x8c, 0x10, 0x8d}  for i = 1,8 do  desTbl.key[i] = 0x33 + i  end    desTbl.flag = 0    local content = {mylib.Des(desTbl)}  LogPrint(LOG\_TYPE.ENUM\_NUMBER,#content,content)  end | |

## 2.4 VerifySignature

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.VerifySignature(sigTbl) | |
| 功能 | | 验证签名 | |
| 参数 | 输入 | 待验证的内容 |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | | local sigTbl =  {  dataLen =0, --数据长度  data = {}, --签名数据  keyLen = 0, --签名公钥长度  key = {}, --签名公钥  hashLen = 0, --签名之前的hash长度  hash = {} --签名hash  } | |
| 返回 | | true：验证成功，false：失败 |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ VerifySignature ()  local sigTbl =  {  dataLen = 9,  data = {0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39},  keyLen = 33,  key = {0x03, 0xee, 0xf7, 0xa3, 0x80, 0xbc, 0xf9, 0xcf, 0x97, 0x5d, 0x91, 0x6f, 0xda, 0xb1, 0x8d, 0x08, 0x1c, 0x9d, 0x55, 0xba, 0x43, 0x46, 0x54, 0x35, 0xa4, 0xd1, 0xcc, 0x59, 0x86, 0x10, 0xa4, 0x44, 0x79},  hashLen = 32,  hash = {0x24, 0x4f, 0xa7, 0xcf, 0x97, 0xae, 0x15, 0x85, 0xd8, 0xf8, 0x02, 0x4b, 0xa1, 0x8b, 0x8a, 0xbe, 0xce, 0x8e, 0xb9, 0xcd, 0x4d, 0x01, 0x6d, 0xd0, 0xba, 0x8c, 0xc0, 0xdc, 0x85, 0x1a, 0x9c, 0x0e}  }    local ret = mylib.VerifySignature(sigTbl)  if ret then  LogPrint(LOG\_TYPE.ENUM\_STRING,string.len("ok"),"ok")  else  LogPrint(LOG\_TYPE.ENUM\_STRING,string.len("bad"),"bad")  end  end | |

## 2.5 GetTxContracts

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetTxContracts(hashTbl) | |
| 功能 | | 获取指定hash的合约内容 | |
| 参数 | 输入 | 合约交易hash |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：返回合约内容；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ GetTxContracts ()  local hash = {0x24, 0x4f, 0xa7, 0xcf, 0x97, 0xae, 0x15, 0x85, 0xd8, 0xf8, 0x02, 0x4b, 0xa1, 0x8b, 0x8a, 0xbe, 0xce, 0x8e, 0xb9, 0xcd, 0x4d, 0x01, 0x6d, 0xd0, 0xba, 0x8c, 0xc0, 0xdc, 0x85, 0x1a, 0x9c, 0x0e}  local content = {mylib.GetTxContracts(Unpack(hash))}  LogPrint(LOG\_TYPE.ENUM\_NUMBER,#content,content)  end | |

## 2.6 GetTxAccounts

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetTxAccounts(hashTbl) | |
| 功能 | | 获取指定hash的交易账户 | |
| 参数 | 输入 | 交易hash |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：返回交易账户；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ GetTxAccounts ()  local hash = {0x24, 0x4f, 0xa7, 0xcf, 0x97, 0xae, 0x15, 0x85, 0xd8, 0xf8, 0x02, 0x4b, 0xa1, 0x8b, 0x8a, 0xbe, 0xce, 0x8e, 0xb9, 0xcd, 0x4d, 0x01, 0x6d, 0xd0, 0xba, 0x8c, 0xc0, 0xdc, 0x85, 0x1a, 0x9c, 0x0e}  local accounts = {mylib.GetTxAccounts(Unpack(hash))}  LogPrint(LOG\_TYPE.ENUM\_NUMBER,# accounts, accounts)  end | |

## 2.7 GetAccountPublickey

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetAccountPublickey(accountTbl[1],accountTbl[2],accountTbl[3],accountTbl[4],accountTbl[5],accountTbl[6]) | |
| 功能 | | 获取账户的公钥 | |
| 参数 | 输入 | local accountTbl = {mylib.GetCurTxAccount()} |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：账户的publickey；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_GetAccountPublickey()  local accountTbl = {mylib.GetCurTxAccount()}  local i  local result = {mylib.GetAccountPublickey(accountTbl[1],accountTbl[2],accountTbl[3],  accountTbl[4],accountTbl[5],accountTbl[6])}  assert(#result == 33,"GetAccountPublickey err");    for i = 1,#result do  print("Publickey",i,(result[i]))  end  end | |

## 2.8 QueryAccountBalance

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.QueryAccountBalance(unpack(accountTbl)) | |
| 功能 | | 获取账户的余额 | |
| 参数 | 输入 | local accountTbl = {mylib.GetCurTxAccount()} |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：账户的余额；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_QueryAccountBalance()  local accountTbl = {mylib.GetCurTxAccount()}  local i  local result = {mylib.QueryAccountBalance(unpack(accountTbl))}  assert(#result == 8,"QueryAccountBalance err");  for i = 1,#result do  print("Balance",i,(result[i]))  end  end | |

## 2.9 GetTxConFirmHeight

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetTxConFirmHeight(unpack(txhashTbl)) | |
| 功能 | | 获取指定hash交易的确认高度 | |
| 参数 | 输入 | 交易hash |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：交易确认高度；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_GetTxConFirmHeight()  -- "hash" :"4a2af2d83683325e780f2b859e7421f4592e3105d01017aab45c15da3910be8e"  local txhashTbl = {0x4a,0x2a,0xf2,0xd8,0x36,0x83,0x32,0x5e,  0x78,0x0f,0x2b,0x85,0x9e,0x74,0x21,0xf4,  0x59,0x2e,0x31,0x05,0xd0,0x10,0x17,0xaa,  0xb4,0x5c,0x15,0xda,0x39,0x10,0xbe,0x8e}  local result = mylib.GetTxConFirmHeight(unpack(txhashTbl))  assert(result > 0,"GetTxConFirmHeight err");  end | |

## 2.10 GetBlockHash(height)

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetBlockHash(height) | |
| 功能 | | 获取指定块hash | |
| 参数 | 输入 | 块的高度 |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：块hash值；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_GetBlockHash()  local height = 47037  local i  local result = {mylib.GetBlockHash(height)}  assert(#result == 32,"GetBlockHash err");  for i = 1,#result do  print("BlockHash",i,(result[i]))  end  end | |

## 2.11 GetCurRunEnvHeight

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetCurRunEnvHeight() | |
| 功能 | | 获取当前运行高度 | |
| 参数 | 输入 |  |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：块的高度；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_GetCurRunEnvHeight()  local result = mylib.GetCurRunEnvHeight()  assert(result > 0,"GetCurRunEnvHeight err");  print("RunEnvHeight",result)  end | |

## 2.12 WriteData

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.WriteData(writeDbTbl) | |
| 功能 | | 写脚本数据库 | |
| 参数 | 输入 | local writeDbTbl = {  key = "config", --关键字  length = 0, -- value数据流的总长  value = {} --值  } |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：执行成功；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_WriteData()  local writeDbTbl = {  key = "config", --关键字  length = 0, -- value数据流的总长  value = {} --值  }  addressTbl = {0x64,0x63,0x6D,0x43,0x62,0x4B,0x62,0x41,  0x66,0x4B,0x72,0x6F,0x66,0x4E,0x7A,0x33,  0x35,0x4D,0x53,0x46,0x75,0x70,0x78,0x72,  0x78,0x34,0x55,0x77,0x6E,0x33,0x76,0x67,  0x6A,0x4C}  --每次限额,每日限额,企业登记的账户地址  writeDbTbl.value = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x64,  0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x2C,  unpack(addressTbl)}  writeDbTbl.length = #writeDbTbl.value  assert(mylib.WriteData(writeDbTbl),"WriteData err")  end | |

## 2.13 DeleteData

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.DeleteData(key) | |
| 功能 | | 删除脚本数据库 | |
| 参数 | 输入 | 关键字key =“config” |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：true；失败：false |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ DeleteData ()  local writeDbTbl = {  key = "config", --关键字  length = 0, -- value数据流的总长  value = {} --值  }  assert(mylib.DeleteData(writeDbTbl.key),"DeleteData err")  print("DeleteData return ok")  local readResult = {}  readResult = {mylib.ReadData(writeDbTbl.key)}  if(TableIsEmpty(readResult)) then  print("DeleteData ok")  end  end | |

## 2.14 ReadData

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.ReadData(key) | |
| 功能 | | 根据指定的key读取脚本数据库中的记录 | |
| 参数 | 输入 | 关键字key =“config” |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：返回value；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ ReadData ()  local key =“config”  local readResult = {mylib.ReadData(key)}  assert(#readResult > 0,"ReadData0 err")  local i  for i = 1,#readResult do  print("",i,(readResult[i]))  end  end | |

## 2.15 ModifyData

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.ModifyData(writeDbTbl) | |
| 功能 | | 修改脚本数据库内容 | |
| 参数 | 输入 | local writeDbTbl = {  key = "config", --关键字  length = 0, -- value数据流的总长  value = {} --值  } |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：True；失败：false |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ ModifyData ()  local writeDbTbl = {  key = "config", --关键字  length = 0, -- value数据流的总长  value = {} --值  }  addressTbl = {0x64,0x63,0x6D,0x43,0x62,0x4B,0x62,0x41,  0x66,0x4B,0x72,0x6F,0x66,0x4E,0x7A,0x33,  0x35,0x4D,0x53,0x46,0x75,0x70,0x78,0x72,  0x78,0x34,0x55,0x77,0x6E,0x33,0x76,0x67,  0x6A,0x4C}  --每次限额,每日限额,企业登记的账户地址  writeDbTbl.value = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x64,  0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x2C,  unpack(addressTbl)}  writeDbTbl.length = #writeDbTbl.value  writeDbTbl.value[8] = 200  assert(mylib.ModifyData(writeDbTbl),"ModifyData err")  print("ModifyData ok")  readResult = {}  readResult = {mylib.ReadData(writeDbTbl.key)}  assert(#readResult > 0,"ReadData1 err")  print("ReadData1 ok")  for i = 1,#readResult do  print("",i,(readResult[i]))  end  end | |

## 2.16 GetCurTxHash

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetCurTxHash() | |
| 功能 | | 获取当前交易hash | |
| 参数 | 输入 |  |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：Hash值；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_GetCurTxHash()  local result = {mylib.GetCurTxHash()}  assert(#result == 32,"GetCurTxHash err");  for i = 1,#result do  print("CurTxHash",i,(result[i]))  end  end | |

## 2.17 WriteOutput

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.WriteOutput(writeOutputTbl) | |
| 功能 | | 操作系统账户 | |
| 参数 | 输入 | local writeOutputTbl = {  addrType = 1, --账户类型 REG\_ID = 0x01,BASE\_58\_ADDR = 0x02,  accountIdTbl = {}, --account id  operatorType = 0, --操作类型  outHeight = 0, --超时高度  moneyTbl = {} --金额  } |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：True；失败：False |  |
|  |  |
| 注释 | |  | |
| 示例 | | function WriteWithDrawal(accTbl,moneyTbl)  --执行系统账户提现操作  assert(TableIsNotEmpty(accTbl),"WriteWithDrawal accTbl invlaid1")  assert(TableIsNotEmpty (moneyTbl),"WriteWithDrawal moneyTbl invlaid1")    local writeOutputTbl = {  addrType = 1, --账户类型 REG\_ID = 0x01,BASE\_58\_ADDR = 0x02,  accountIdTbl = {}, --account id  operatorType = 0, --操作类型  outHeight = 0, --超时高度  moneyTbl = {} --金额  }    writeOutputTbl.accountIdTbl = {unpack(accTbl)}  writeOutputTbl.operatorType = OPER\_TYPE. ENUM\_ADD\_FREE  writeOutputTbl.moneyTbl = {unpack(moneyTbl)}  assert(mylib.WriteOutput(writeOutputTbl),"WriteWithDrawal WriteOutput err0")    writeOutputTbl.operatorType = OPER\_TYPE. ENUM\_MINUS\_FREE  writeOutputTbl.accountidTbl = {mylib.GetScriptID()}  assert(mylib.WriteOutput(writeOutputTbl),"WriteWithDrawal WriteOutput err1")  end | |

## 2.18 GetScriptData

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetScriptData(paraTbl) | |
| 功能 | | 获取脚本数据 | |
| 参数 | 输入 | local paraTbl = {  id = {0x00,0x01,0x00,0x00,0xb7,0xc4}, --6字节的脚本id  key = "config", --关键字  } |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：脚本数据；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_GetScriptData()  mylib\_WriteData()  local paraTbl = {  --id = {0x00,0x01,0x00,0x00,0xb7,0xc4}, --6字节的脚本id  id = {mylib.GetScriptID()},  key = "config", --关键字  }  local result = {mylib.GetScriptData(paraTbl)}  assert(#result > 0,"GetScriptData err")  for i = 1,#result do  print("GetScriptData",i,(result[i]))  end  end | |

## 2.19 GetScriptID

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetScriptID() | |
| 功能 | | 获取脚本ID | |
| 参数 | 输入 |  |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：返回6字节的脚本ID；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_GetScriptID()  local result = {mylib. GetScriptID ())}  assert(#result > 0," GetScriptID err")  for i = 1,#result do  print("GetScriptID",i,(result[i]))  end  end | |

## 2.20 GetCurTxAccount

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetCurTxAccount() | |
| 功能 | | 获取当前交易账户 | |
| 参数 | 输入 |  |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：返回账户信息；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ GetCurTxAccount ()  local i  local result = {mylib. GetCurTxAccount ())}  assert(#result == 6," GetCurTxAccount err");  for i = 1,#result do  print("Account",i,(result[i]))  end  end | |

## 2.21 GetCurTxPayAmount

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetCurTxPayAmount() | |
| 功能 | | 获取当前交易金额 | |
| 参数 | 输入 |  |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：返回Int64的交易金额；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ GetCurTxPayAmount ()  local i  local paymoneyTbl = {mylib. GetCurTxPayAmount ())}  assert(#paymoneyTbl == 8," GetCurTxPayAmount err");  for i = 1,# paymoneyTbl do  print("PayAmount ",i,( paymoneyTbl [i]))  end  end | |

## 2.22 GetUserAppAccValue

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetUserAppAccValue(idTbl) | |
| 功能 | | 获取指定账户的余额 | |
| 参数 | 输入 | -- 用户id  local idTbl = {  idlen = 0, --id长度  idValueTbl = {} --id值  } |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：返回Int64的交易金额；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ GetUserAppAccValue ()  local idTbl = {  idlen = 6, --id长度  idValueTbl = {0x01,0x02,0x03,0x04,0x05,0x06} --id值  }  local i  local money = { mylib.GetUserAppAccValue(idTbl) }  assert(#money == 8," GetUserAppAccValue err");  for i = 1,# money do  print("money ",i,( money [i]))  end  end | |

## 2.23 GetUserAppAccFoudWithTag

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetUserAppAccFoudWithTag(app\_operateTbl) | |
| 功能 | | 获取指定账户的余额 | |
| 参数 | 输入 | -- 用户id  local idTbl = {  idlen = 0, --id长度  idValueTbl = {} --id值  } |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：返回Int64的交易金额；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ GetUserAppAccValue ()  local idTbl = {  idlen = 6, --id长度  idValueTbl = {0x01,0x02,0x03,0x04,0x05,0x06} --id值  }  local i  local money = { mylib.GetUserAppAccValue(idTbl) }  assert(#money == 8," GetUserAppAccValue err");  for i = 1,# money do  print("money ",i,( money [i]))  end  end | |

## 2.24 WriteOutAppOperate

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.WriteOutAppOperate(app\_operateTbl) | |
| 功能 | | 操作系统应用账户 | |
| 参数 | 输入 | local app\_operateTbl = {  operatorType = 0, --操作类型  outHeight = 0, --超时高度  moneyTbl = {}, --金额  userIdLen = 0, --账户ID长度  userIdTbl = {}, --账户ID  fundTagLen = 0, --fund tag len  fundTagTbl = {} --fund tag  } |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：True；失败：False |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ WriteOutAppOperate ()  local app\_operateTbl = {  operatorType = 0, --操作类型  outHeight = 0, --超时高度  moneyTbl = {}, --金额  userIdLen = 0, --账户ID长度  userIdTbl = {}, --账户ID  fundTagLen = 0, --fund tag len  fundTagTbl = {} --fund tag  }  app\_operateTbl.operatorType = AppOperatorTable. ENUM\_ADD\_FREE\_OP  app\_operateTbl.moneyTbl = {0,0,0,0,0,0,1,44}  app\_operateTbl.userIdLen = 6  app\_operateTbl.userIdTbl = {0,0,0,0,2,0}  assert(mylib.WriteOutAppOperate(app\_operateTbl),"WriteOutAppOperate err0")  end | |

## 2.25 GetBase58Addr

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.GetBase58Addr(unpack(accountTbl)) | |
| 功能 | | 获取指定账户的base58地址 | |
| 参数 | 输入 | local accountTbl = {5,157,0,0,7,34} --6字节的账户ID |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：base58地址；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_GetBase58Addr()  local accountTbl = {5,157,0,0,7,34} --6字节的账户ID    local result = {mylib.GetBase58Addr(unpack(accountTbl))}  assert(#result > 0,"GetBase58Addr err")  if(#result > 0) then  LogPrint(LOG\_TYPE.NUMBER,#result,result)  end  end | |

## 2.26 ByteToInteger

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib. ByteToInteger (byteTbl[1], byteTbl [2], byteTbl [3], byteTbl [4]) | |
| 功能 | | 将字节转换成number | |
| 参数 | 输入 | 4字节流或8字节的字节流 |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：Integer高度值或金额值；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_ByteToInteger()  local height = mylib.ByteToInteger(0xa0,0x05,0x00,0x00)  assert(height > 0,"ByteToInteger error0")  print("height=",height)    local money = mylib.ByteToInteger(0x78,0x56,0x34,0x12,0x78,0x56,0x34,0x12)  assert(money > 0,"ByteToInteger error1")  print("money=",money)  end | |

## 2.27 IntegerToByte4

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib. IntegerToByte4 (height) | |
| 功能 | | 将Interger的高度值转换成4字节 | |
| 参数 | 输入 | Integer高度值 |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：返回4个字节；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_IntegerToByte()  local height = 1440  local result = {mylib.IntegerToByte4(height)}  assert(#result == 4,"IntegerToByte4 error0")  local i  print("height byte")  for i = 1,#result do  print("",i,(result[i]))  end  end | |

## 2.28 IntegerToByte8

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib. IntegerToByte8 (money) | |
| 功能 | | 将Integer的高度值转换成8字节 | |
| 参数 | 输入 | Integer金额值 |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：返回8个字节；失败：nil |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_IntegerToByte8()  local money = 1311768465173141112  local result = {mylib.IntegerToByte8(money)}  assert(#result == 8,"IntegerToByte8 error0")  for i = 1,#result do  print("",i,(result[i]))  end  end | |

## 2.29 TransferContactAsset

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib. TransferContactAsset (Unpack(addrTbl)) | |
| 功能 | | 转移全部资产 | |
| 参数 | 输入 | 账户地址 |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：True；失败：False |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_TransferContactAsset()  local accountTbl = {5,157,0,0,7,34} --6字节的账户ID    local addrTbl = {mylib.GetBase58Addr(unpack(accountTbl))}  assert(#addrTbl > 0,"GetBase58Addr err")  assert(mylib.TransferContactAsset(Unpack(addrTbl)), "TransferContactAsset err")  end | |

## 2.30 TransferSomeAsset

|  |  |  |  |
| --- | --- | --- | --- |
| 原型 | | mylib.TransferSomeAsset (assetOperateTbl) | |
| 功能 | | 转移部分资产 | |
| 参数 | 输入 | local assetOperateTbl = {  toAddrTbl = {}, --转移目的地址  outHeight = 0, --高度  moneyTbl = {}, --资产数  fundTagLen = 0, --fund tag len  fundTagTbl = {} --fund tag  } |  |
| 输出 |  | |
| 常量 | |  | |
| 结构体 | |  | |
| 返回 | | 成功：True；失败：False |  |
|  |  |
| 注释 | |  | |
| 示例 | | function mylib\_TransferSomeAsset()  local accountTbl = {5,157,0,0,7,34} --6字节的账户ID    local addrTbl = {mylib.GetBase58Addr(unpack(accountTbl))}  assert(#addrTbl > 0,"GetBase58Addr err")  local assetOperateTbl = {  toAddrTbl = {}, --转移地址  outHeight = 0, --高度  moneyTbl = {}, --资产数  fundTagLen = 0, --fund tag len  fundTagTbl = {} --fund tag  }  local money = 1311768465173141112  local moneyTbl = {mylib.IntegerToByte8(money)}  assetOperateTbl.toAddrTbl = addrTbl  assetOperateTbl.outHeight = 1440  assetOperateTbl.moneyTbl = moneyTbl  assert(mylib.TransferSomeAsset(assetOperateTbl), "TransferSomeAsset err")  end | |

# 注意事项

## 3.1由于安全性考虑，合约中只能使用lua中table，math，string库，不能使用io操作等。

## 3.2脚本现在只能写在一个文件中，不支持多文件。

## 3.3脚本大小不能超过64k。

## 3.4合约内容通过全局变量“contract”传给脚本，脚本里面可以拿来就用

## 3.5账户平衡检测开关通过在lua脚本中的全局变量“gCheckAccount”控制，默认关闭此开关。

## 3.6合约内容不能超过4096字节。

## 3.7脚本中如果有异常，或逻辑不对，可以通过assert，error退出执行，lua解释器会捕获到该异常。

## 3.8脚本中必须以mylib = require "mylib"开头，通过mylib调用里面的API。

# 示例 （充值和提现操作）

mylib = require "mylib"

--日志类型

LOG\_TYPE =

{

ENUM\_STRING = 0, --字符串类型

ENUM\_NUMBER = 1, --数字类型

};

--系统账户操作定义

OPER\_TYPE =

{

ENUM\_ADD\_FREE = 1, --系统账户加

ENUM\_MINUS\_FREE = 2 --系统账户减

}

--脚本应用账户操作类型定义

APP\_OPERATOR\_TYPE =

{

ENUM\_ADD\_FREE\_OP = 1, --自由账户加

ENUM\_SUB\_FREE\_OP = 2, --自由账户减

ENUM\_ADD\_FREEZED\_OP = 3, --冻结账户加

ENUM\_SUB\_FREEZED\_OP = 4 --冻结账户减

}

--账户类型

ADDR\_TYPE =

{

ENUM\_REGID = 1, -- REG\_ID

ENUM\_BASE58 = 2, -- BASE58 ADDR

}

--交易类型

TX\_TYPE =

{

TX\_WITHDRAW = 1, --提现

TX\_RECHARGE= 2, --充值

}

FREEZE\_MONTH\_NUM = 20 -- 冻结次数

FREEZE\_PERIOD = 5 -- 冻结周期

MAX\_MONEY = 100000000000000000 -- 总金额限制

FREE\_MONEY = 10000000000000000 -- 自由金额限制

gCheckAccount = true -- 平衡检查 true 打开，false 关闭

gSendAccountTbl = -- 发币地址

{

0x00, 0x00, 0x00, 0x00, 0x01, 0x00

}

--判断表是否为空

function TableIsEmpty(t)

return \_G.next(t) == nil

end

--判断表非空

function TableIsNotEmpty(t)

return false == TableIsEmpty(t)

end

--[[

功能:日志输出

参数：

aKey:日志类型

bLength:日志长度

cValue：日志

--]]

function LogPrint(aKey,bLength,cValue)

assert(bLength >= 1,"LogPrint bLength invlaid")

if(aKey == LOG\_TYPE.ENUM\_STRING) then

assert(type(cValue) == "string","LogPrint cValue invlaid0")

elseif(aKey == LOG\_TYPE.ENUM\_NUMBER) then

assert(TableIsNotEmpty(cValue),"LogPrint cValue invlaid1")

else

error("LogPrint aKey invlaid")

end

local logTable = {

key = LOG\_TYPE.ENUM\_STRING,

length = 0,

value = nil

}

logTable.key = aKey

logTable.length = bLength

logTable.value = cValue

mylib.LogPrint(logTable)

end

--[[

功能:遍历表元素

参数：

t:表

i:开始索引

--]]

function Unpack(t,i)

i = i or 1

if t[i] then

return t[i],Unpack(t,i+1)

end

end

--[[

功能:比较两表元素是否相同

参数：

tDest:表1

tSrc:表2

start1:开始索引

返回值：

0:相等

1:表1>表2

2:表1<表2

--]]

function MemCmp(tDest,tSrc,start1)

assert(TableIsNotEmpty(tDest),"tDest is empty")

assert(TableIsNotEmpty(tSrc),"tSrc is empty")

local i

for i = #tDest,1,-1 do

if tDest[i] > tSrc[i + start1 - 1] then

return 1

elseif tDest[i] < tSrc[i + start1 - 1] then

return -1

else

end

end

return 0

end

--[[

功能:获取表从开始索引指定长度的元素集合

参数：

tbl:表

start:开始索引

length:长度

返回值：

一个新表

--]]

function GetValueFromContract(tbl,start,length)

assert(start > 0,"GetValueFromContract start err")

assert(length > 0,"GetValueFromContract length err")

local newTab = {}

local i

for i = 0,length -1 do

newTab[1 + i] = tbl[start + i]

end

return newTab

end

--[[

功能:充值

参数：

无

返回值：

true:成功

false:失败

流程：

1）获取当前交易账户，并判断是否与发币地址一致

2）获取合约相应内容，并判断合法性

3）将自由金额提现到系统账户下

4）冻结相应的冻结部分

5）扣除相应的脚本账户下的金额

--]]

function Recharge()

-- 1）

-- local accountTbl = {0, 0, 0, 0, 0, 0}

-- accountTbl = {mylib.GetCurTxAccount()}

-- assert(TableIsNotEmpty(accountTbl),"GetCurTxAccount error")

-- assert(MemCmp(gSendAccountTbl, accountTbl,1) == 0,"Recharge address err")

-- 2）

local toAddrTbl = {}

toAddrTbl = GetValueFromContract(contract, 3, 34)

local moneyTbl = {}

moneyTbl = GetValueFromContract(contract, 37, 8)

local money = mylib.ByteToInteger(Unpack(moneyTbl))

assert(money > 0,"money <= 0")

local freeMoneyTbl = {}

freeMoneyTbl = GetValueFromContract(contract, 45, 8)

local freeMoney = mylib.ByteToInteger(Unpack(freeMoneyTbl))

assert(freeMoney > 0,"freeMoney <= 0")

local freeMonthMoneyTbl = {}

freeMonthMoneyTbl = GetValueFromContract(contract, 53, 8)

local freeMonthMoney = mylib.ByteToInteger(Unpack(freeMonthMoneyTbl))

assert(freeMonthMoney > 0,"freeMonthMoney <= 0")

local payMoneyTbl = {}

payMoneyTbl = {mylib.GetCurTxPayAmount()}

assert(TableIsNotEmpty(payMoneyTbl),"GetCurTxPayAmount error")

local payMoney = mylib.ByteToInteger(Unpack(payMoneyTbl))

assert(payMoney > 0,"payMoney <= 0")

-- 总金额与充值金额要相等

assert(money == payMoney, "充值金额不正确1")

-- 总金额不能小于1,不能小于自由金额或月冻结金额

if money < 1

or money < freeMoney

or money < freeMonthMoney then

LogPrint(LOG\_TYPE.ENUM\_STRING, string.len("充值金额不正确2"), "充值金额不正确2");

error("充值金额不正确2")

end

-- 三个金额，上限值检测

if money >= MAX\_MONEY

or freeMoney >= FREE\_MONEY

or freeMonthMoney >= FREE\_MONEY then

LogPrint(LOG\_TYPE.ENUM\_STRING, string.len("充值金额不正确3"), "充值金额不正确3");

error("充值金额不正确3")

end

local freezeNum = FREEZE\_MONTH\_NUM - 1

assert(freezeNum > 0, "月冻结总数不正确")

-- 总金额不能小于总的月冻结金额

local freezeMoney = freeMonthMoney \* freezeNum

if freezeMoney < freezeNum

or freezeMoney < freeMoney

or money < freezeMoney then

LogPrint(LOG\_TYPE.ENUM\_STRING, string.len("充值金额不正确4"), "充值金额不正确4");

error("充值金额不正确4")

end

-- 检查充值总金额和自由金额加上冻结金额是否相等

freezeMoney = freezeMoney + freeMoney

if money ~= freezeMoney then

LogPrint(LOG\_TYPE.ENUM\_STRING, string.len("充值金额不正确5"), "充值金额不正确5");

error("充值金额不正确5")

end

-- 3）

--操作系统账户的结构

local writeOutputTbl =

{

addrType = 1, --账户类型 REG\_ID = 0x01,BASE\_58\_ADDR = 0x02,

accountIdTbl = {}, --account id

operatorType = 0, --操作类型

outHeight = 0, --超时高度

moneyTbl = {} --金额

}

writeOutputTbl.addrType = ADDR\_TYPE.ENUM\_BASE58

writeOutputTbl.operatorType = OPER\_TYPE.ENUM\_ADD\_FREE

writeOutputTbl.accountIdTbl = {Unpack(toAddrTbl)}

writeOutputTbl.moneyTbl = {Unpack(freeMoneyTbl)}

assert(mylib.WriteOutput(writeOutputTbl),"WriteOutput err1")

local curHeight = 0

curHeight = mylib.GetCurRunEnvHeight()

-- 4）

local appOperateTbl = {

operatorType = 0, -- 操作类型

outHeight = 0, -- 超时高度

moneyTbl = {},

userIdLen = 0, -- 地址长度

userIdTbl = {}, -- 地址

fundTagLen = 0, -- fund tag len

fundTagTbl = {} -- fund tag

}

appOperateTbl.operatorType = APP\_OPERATOR\_TYPE.ENUM\_ADD\_FREEZED\_OP

appOperateTbl.userIdLen = 34

appOperateTbl.userIdTbl = toAddrTbl

appOperateTbl.moneyTbl = freeMonthMoneyTbl

for i = 1, freezeNum do

appOperateTbl.outHeight = curHeight + FREEZE\_PERIOD \* i

assert(mylib.WriteOutAppOperate(appOperateTbl),"WriteOutAppOperate err1")

end

-- 5）

writeOutputTbl.addrType = ADDR\_TYPE.ENUM\_REGID

writeOutputTbl.operatorType = OPER\_TYPE.ENUM\_MINUS\_FREE

writeOutputTbl.accountIdTbl = {mylib.GetScriptID()}

assert(mylib.WriteOutput(writeOutputTbl),"WriteOutput err2")

return true

end

--[[

功能:操作系统账户

参数：

accTbl:账户

moneyTbl:操作金额

返回值：

true:成功

false:失败

流程：

1）增加该系统账户金额

2）相应地减少脚本账户金额

--]]

function WriteWithdrawal(accTbl,moneyTbl)

--操作系统账户的结构

local writeOutputTbl =

{

addrType = 1, --账户类型 REG\_ID = 0x01,BASE\_58\_ADDR = 0x02,

accountIdTbl = {}, --account id

operatorType = 0, --操作类型

outHeight = 0, --超时高度

moneyTbl = {} --金额

}

--执行系统账户提现操作

assert(TableIsNotEmpty(accTbl),"WriteWithDrawal accTbl invlaid1")

assert(TableIsNotEmpty(moneyTbl),"WriteWithDrawal moneyTbl invlaid1")

-- 1）

writeOutputTbl.addrType = ADDR\_TYPE.ENUM\_REGID

writeOutputTbl.operatorType = OPER\_TYPE.ENUM\_ADD\_FREE

writeOutputTbl.accountIdTbl = {Unpack(accTbl)}

writeOutputTbl.moneyTbl = {Unpack(moneyTbl)}

assert(mylib.WriteOutput(writeOutputTbl),"WriteWithDrawal WriteOutput err0")

-- 2）

writeOutputTbl.operatorType = OPER\_TYPE.ENUM\_MINUS\_FREE

writeOutputTbl.accountIdTbl = {mylib.GetScriptID()}

assert(mylib.WriteOutput(writeOutputTbl),"WriteWithDrawal WriteOutput err1")

return true

end

--[[

功能:提现

参数：

addrType:账户类型，BASE58或REGID

返回值：

true:成功

false:失败

流程：

1）判断账户类型

2）获取当前交易账户

3）获取该账户下的自由金额

4）减掉该账户下的自由金额

5）相应的操作系统账户

--]]

function Withdraw(addrType)

-- 1）

if addrType ~= ADDR\_TYPE.ENUM\_REGID and addrType ~= ADDR\_TYPE.ENUM\_BASE58 then

error("In function Withdraw, addr type err")

end

-- 2）

local accountTbl = {0, 0, 0, 0, 0, 0}

accountTbl = {mylib.GetCurTxAccount()}

assert(TableIsNotEmpty(accountTbl),"GetCurTxAccount error")

local idTbl =

{

idLen = 0,

idValueTbl = {}

}

if addrType == ADDR\_TYPE.ENUM\_REGID then

idTbl.idLen = 6

idTbl.idValueTbl = accountTbl

else

local base58Addr = {}

base58Addr = {mylib.GetBase58Addr(Unpack(accountTbl))}

assert(TableIsNotEmpty(base58Addr),"GetBase58Addr error")

idTbl.idLen = 34

idTbl.idValueTbl = base58Addr

end

-- 3）

local freeMoneyTbl = {mylib.GetUserAppAccValue(idTbl)}

assert(TableIsNotEmpty(freeMoneyTbl),"GetUserAppAccValue error")

local freeMoney = mylib.ByteToInteger(Unpack(freeMoneyTbl))

assert(freeMoney > 0,"Account balance is 0.")

-- 4）

local appOperateTbl = {

operatorType = 0, -- 操作类型

outHeight = 0, -- 超时高度

moneyTbl = {},

userIdLen = 0, -- 地址长度

userIdTbl = {}, -- 地址

fundTagLen = 0, -- fund tag len

fundTagTbl = {} -- fund tag

}

appOperateTbl.operatorType = APP\_OPERATOR\_TYPE.ENUM\_SUB\_FREE\_OP

appOperateTbl.userIdLen = idTbl.idLen

appOperateTbl.userIdTbl = idTbl.idValueTbl

appOperateTbl.moneyTbl = freeMoneyTbl

assert(mylib.WriteOutAppOperate(appOperateTbl),"WriteOutAppOperate err1")

-- 5）

assert(WriteWithdrawal(accountTbl, freeMoneyTbl), "WriteWithdrawal err")

return true

end

function Main()

--[[

local i = 1

for i = 1,#contract do

print("contract")

print(" ",i,(contract[i]))

end

--]]

assert(#contract >= 2,"contract length err.")

assert(contract[1] == 0xff,"Contract identification error.")

if contract[2] == TX\_TYPE.TX\_RECHARGE then

assert(#contract == 60,"recharge contract length err.")

Recharge()

elseif contract[2] == TX\_TYPE.TX\_WITHDRAW then

assert(#contract == 11,"withdraw contract length err.")

Withdraw(contract[3])

else

error("RUN\_SCRIPT\_DATA\_ERR")

end

end

Main()

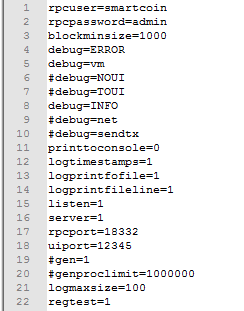
# 脚本调试

以上面的示例代码为例，将其代码保存为test.lua，文件格式保存为utf-8：

## 5.1 启动dacrs程序

最好在linux下测试，windows上编译，调试慢。调试时先在regtest下调试通过，然后才在testnet和main网络上注册。

修改Dacrs.conf文件，添加regtest=1，打开vm调试开关，debug=vm，注释掉gen=1，genproclimit=1000000，采用手动挖矿。



下面所用到的RPC命令，请参考RPC接口文档《RPCCommandV0.1 beta.docx》。

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test

Dacrs version v1.1.2.5-8a88c65-dirty-release-linux (2016-10-11 10:03:03 +0800)

Using OpenSSL version OpenSSL 1.0.1f 6 Jan 2014

Startup time: 2016-10-11 08:14:26

Default data directory /home/xgc/.Dacrs

Using data directory /home/share/xgc/dacrs\_test/regtest

Using at most 125 connections ( 1024 file descriptors available)

## 5.2 导入私钥

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test importprivkey cNcJkU44oG3etbWoEvY46i5qWPeE8jVb7K44keXxEQxsXUZ85MKU

{

"imorpt key address" : "dk2NNjraSvquD9b4SQbysVRQeFikA55HLi"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test getaccountinfo dk2NNjraSvquD9b4SQbysVRQeFikA55HLi

{

"Address" : "dk2NNjraSvquD9b4SQbysVRQeFikA55HLi",

"KeyID" : "4fb64dd1d825bb6812a7090a1d0dd2c75b55242e",

"RegID" : "0-20",

"PublicKey" : "03ae28a4100145a4c354338c727a54800dc540069fa2f5fd5d4a1c80b4a35a1762",

"MinerPKey" : "",

"Balance" : 100000082000000000,

"CoinDays" : 0,

"UpdateHeight" : 263,

"CurCoinDays" : 694445,

"postion" : "inblock"

}

## 5.3 生成几个测试地址，并充值激活

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test getnewaddress

{

"addr" : "druMpuM8YEkvzyErLMUgwasyafoficmK7S",

"minerpubkey" : "no"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test getnewaddress

{

"addr" : "dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6",

"minerpubkey" : "no"

}

充币：

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test sendtoaddress druMpuM8YEkvzyErLMUgwasyafoficmK7S 100000000000

{

"hash" : "59b55b1abe2a6aa38d85301158d486ffa367f349af9c18506a4adadc1d999a41"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test sendtoaddress dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6 100000000000

{

"hash" : "e6ac958bbc202cb300bbfcb21ed66e6279f009128ec17e9354796e3d12ae5dd5"

}

手动挖矿，看生成的两地址里现在是否有币

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test setgenerate true 1

{

"msg" : "in mining"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test getbalance dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6

{

"balance" : 1000.00000000

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test getbalance druMpuM8YEkvzyErLMUgwasyafoficmK7S

{

"balance" : 1000.00000000

}

激活这两个地址

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test registaccounttx dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6 100000

{

"hash" : "02d86cdecb3cb9a9d92bd28cd6023c58f771bd3df03aa0689e91b26c2307a976"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test registaccounttx druMpuM8YEkvzyErLMUgwasyafoficmK7S 100000

{

"hash" : "6638af4edc5e417d8abd214bb6c1db1f41f83ea2e4b8c5924d2bcebfe4e25741"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test setgenerate true 1

{

"msg" : "in mining"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test getaccountinfo dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6

{

"Address" : "dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6",

"KeyID" : "8dfa4bec33ea7ce2692a6dc4ea3e47e1085cc0b7",

"RegID" : "293-1",

"PublicKey" : "0208f52f6e296805bb8aa44a3c44454cc15ed5abca43b9f95ae73f1ef2c99f2bc1",

"MinerPKey" : "",

"Balance" : 99999900000,

"CoinDays" : 9,

"UpdateHeight" : 293,

"CurCoinDays" : 9,

"postion" : "inblock"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test getaccountinfo druMpuM8YEkvzyErLMUgwasyafoficmK7S

{

"Address" : "druMpuM8YEkvzyErLMUgwasyafoficmK7S",

"KeyID" : "9b2bd08576843e5416c7a90739b8ea50d4e98b4f",

"RegID" : "293-2",

"PublicKey" : "027826fbe5ce964c4bd1d55217d103f21214ef603da24224b9d4eee182a2353d32",

"MinerPKey" : "",

"Balance" : 99999900000,

"CoinDays" : 9,

"UpdateHeight" : 293,

"CurCoinDays" : 9,

"postion" : "inblock"

}

RegID字段分别为293-1,293-2,表示地址已经激活。

## 5.4 注册应用脚本

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test registerapptx druMpuM8YEkvzyErLMUgwasyafoficmK7S /home/share/xgc/dacrs\_test/data/test.lua 110000000 0 "test"

{

"hash" : "b9e8906135e64eec879a0dc38079031346dccb0ba273b5503dcd7f3bb0e42e6f"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test setgenerate true 1

{

"msg" : "in mining"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test getscriptid b9e8906135e64eec879a0dc38079031346dccb0ba273b5503dcd7f3bb0e42e6f

{

"regid:" : "295-1",

"script" : "270100000100"

}

获取到脚本appid为295-1

## 5.5 充值操作

合约内容=前缀1字节（0xff）+ 操作类型1字节（0x02）+ 充值地址34字节（dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6）+ 充值总金额 8字节（10000000000）+ 自由金额 8字节（500000000）+ 每月冻结金额 8字节（500000000）

将合约内容转换成16进制字符串

前缀（0xff）=> ff

操作类型（0x02）=> 02

充值地址（dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6）=>

d的ascii码10进制是100，转成16进制是64

q的ascii码10进制是113，转成16进制是71

依次转换，最终结果为6471686269716a705436376f555832615458753366416362707670634a4556345936

充值总金额（10000000000）=>

利用计算器转成16进制为2540be400，补齐8字节后为00 00 00 02 54 0b e4 00，按照内存中逆序后为00e40b5402000000

自由金额（500000000）=>

利用计算器转成16进制为1dcd6500，补齐8字节后为00 00 00 00 1d cd 65 00，按照内存中逆序后为0065cd1d00000000

每月冻结金额（500000000）=>

利用计算器转成16进制为1dcd6500，补齐8字节后为00 00 00 00 1d cd 65 00，按照内存中逆序后为0065cd1d00000000

将这些字段组合在一起，形成合约内容：

ff026471686269716a705436376f555832615458753366416362707670634a455634593600e40b54020000000065cd1d000000000065cd1d00000000

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test createcontracttx druMpuM8YEkvzyErLMUgwasyafoficmK7S 295-1 10000000000 ff026471686269716a705436376f555832615458753366416362707670634a455634593600e40b54020000000065cd1d000000000065cd1d00000000 100000 0

{

"hash" : "fadb05dca69b2a9a5ee771ec6533b2b856950868af8589476e920c2eea5a465c"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test setgenerate true 1

{

"msg" : "in mining"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test getappaccinfo 295-1 dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6

{

"mAccUserID" : "6471686269716a705436376f555832615458753366416362707670634a4556345936",

"FreeValues" : 0,

"vFreezedFund" : [

{

"value" : 500000000,

"nHeight" : 309,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 314,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 319,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 324,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 329,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 334,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 339,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 344,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 349,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 354,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 359,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 364,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 369,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 374,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 379,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 384,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 389,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 394,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 399,

"vTag" : ""

}

]

}

手动多挖几次矿，等自由金额不为0时，可以提现操作

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test getappaccinfo 295-1 dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6

{

"mAccUserID" : "6471686269716a705436376f555832615458753366416362707670634a4556345936",

"FreeValues" : 500000000,

"vFreezedFund" : [

{

"value" : 500000000,

"nHeight" : 314,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 319,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 324,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 329,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 334,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 339,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 344,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 349,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 354,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 359,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 364,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 369,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 374,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 379,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 384,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 389,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 394,

"vTag" : ""

},

{

"value" : 500000000,

"nHeight" : 399,

"vTag" : ""

}

]

}

## 5.6 提现操作

将自由金额（5个币），提现到地址dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6

合约内容=前缀1字节（0xff）+ 操作类型1字节（0x01）+ 账户类型1字节（0x02）+ 提现金额 8字节（500000000）

将合约内容转换成16进制字符串

前缀（0xff）=> ff

操作类型（0x02）=> 01

账户类型（0x02）=> 02

提现金额（500000000）=>

利用计算器转成16进制为1dcd6500，补齐8字节后为00 00 00 00 1d cd 65 00，按照内存中逆序后为0065cd1d00000000

将这些字段组合在一起，形成合约内容：

ff01020065cd1d00000000

### 5.5.1查看dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6当前余额 ：

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test getbalance dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6

{

"balance" : 1004.99900000

}

### 5.5.2提现

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test createcontracttx dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6 295-1 0 ff01020065cd1d00000000 100000 0

{

"hash" : "ce0cf0817e91780b1f7fa1b63ce0af6cbd788024f0c3c15080c2b6748f12e144"

}

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test setgenerate true 1

{

"msg" : "in mining"

}

### 5.5.3 再次查看dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6当前余额 ，发现已经提现了5个币。

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test getbalance dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6

{

"balance" : 1009.99800000

}

## 5.7 脚本出错处理

假如现在再次提现一次，因为dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6脚本账号下自由金额为0，故提现操作会失败，要查看出错信息，修改Dacrs.conf文件，打开vm调试开关，debug=vm。

排错主要靠LogPrint，assert，error打印出错信息到日志。

xgc@rangershi-MS-7817:/home/share/xgc/dacrs/src$ ./dacrs-d -datadir=/home/share/xgc/dacrs\_test createcontracttx dqhbiqjpT67oUX2aTXu3fAcbpvpcJEV4Y6 295-1 0 ff01020065cd1d00000000 100000 0

error: {"code":-4,"message":"Error:run-script-error"}

打开regtest目录下的vm.log:

2016-10-12 02:12:49 [txmempool.cpp:125]vm: tx hash=69a787e4c7a35f40eff5f696c9f361972a9cb8a7e1e4263e527e689b9bd23e2c CheckTxInMemPool run contract

2016-10-12 02:12:49 [vm/vmrunevn.cpp:125]vm: tx hash:69a787e4c7a35f40eff5f696c9f361972a9cb8a7e1e4263e527e689b9bd23e2c fees=100000 fuelrate=100 maxstep:90000

2016-10-12 02:12:49 [vm/vmrunevn.cpp:93]vm: CVmScriptRun::intial() LUA

2016-10-12 02:12:49 [vm/vmlua.cpp:252]vm: pVmScriptRun=0x7fa006ffb480

2016-10-12 02:12:49 [vm/vmlua.cpp:259]vm: luaL\_loadbuffer fail:[string "line"]:375: Account balance is 0.

2016-10-12 02:12:49 [vm/vmlua.cpp:262]vm: run step=-1

2016-10-12 02:12:49 [vm/vmrunevn.cpp:136]vm: CVmScriptRun::run() LUA

其中luaL\_loadbuffer fail:[string "line"]:375: Account balance is 0.提示了脚本375行账户余额为0，从而知道合约为啥会执行不成功。

如果在开发过程中，边写代码边调试，可能写的代码有语法错误，在执行时也会出错，通过上面的方法可以查看出错误行。将错误修正后，要重新注册该脚本，再进行调试。

# 部署

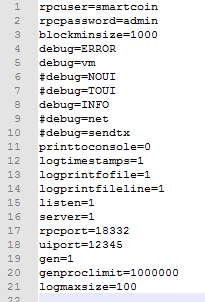
在局域网测试完毕后，可以先在测试网络再测试一遍，然后部署到主网络。

修改配置文件Dacrs.conf。

测试网络配置文件修改：



主网络配置文件修改：



然后像上面那样启动dacrs-d，再使用rpc命令registerapptx注册脚本。