

neo-cg

1、NEO-CLI 命令参考

NEO-CLI 命令参考

打开命令行，定位到 NEO-CLI 所在目录，输入下面代码即可启动 NEO 的命令行钱包（即 NEO 节点）。

```
dotnet neo-cli.dll
```

本篇教程将介绍命令行钱包的所有命令，你可以通过输入命令的形式操作钱包，如创建打开钱包、导入导出私钥、转账、启动共识等。

首先我们先了解一下该命令行钱包有哪些命令，在命令行中输入 `help`，回车，你会看到如图所示的命令列表。

```
neo> help
Normal Commands:
    version
    help
    clear
    exit
Wallet Commands:
    create wallet <path>
    open wallet <path>
    upgrade wallet <path>
    rebuild index
    list address
    list asset
    list key
    show utxo [id|alias]
    show gas
    claim gas
    create address [n=1]
    import key <wif|path>
    export key [address] [path]
    send <id|alias> <address> <value>|all [fee=0]
Node Commands:
    show state
    show node
    show pool [verbose]
    export blocks [path=chain.acc]
Advanced Commands:
    start consensus
    refresh policy
neo> _
```

下面是所有命令的说明，命令中尖括号 <> 表示的是参数，方括号 [] 表示的是可选参数，或符号 | 表示所填的参数可以是其中任意一种，等号 = 表示可选参数在不输入情况下的默认值。

快速同步区块数据

客户端运行时会自动同步区块数据，打开钱包时也会自动同步钱包数据，当同步完成后才可以正常使用客户端以及查看钱包内资产。由于区块链数据庞大，初次同步时等待时间通常很久，建议采用离线同步包进行同步，相关信息，请参见 [快速同步区块数据](#)。

控制台指令

命令	功能说明
version	显示当前软件的版本
help	帮助菜单
clear	清除屏幕
exit	退出程序

钱包操作

命令	功能说明	备注
create wallet \	创建钱包文件	
open wallet \	打开钱包文件	
upgrade wallet \	升级旧版钱包文件	
rebuild index	重建钱包索引	需要打开钱包
list address	列出钱包中的所有账户	需要打开钱包
list asset	列出钱包中的所有资产	需要打开钱包
list key	列出钱包中的所有公钥	需要打开钱包
show utxo [id alias]	列出钱包中指定资产的 UTXO	需要打开钱包
show gas	列出钱包中的所有可提取及不可提取的 GAS	需要打开钱包
claim gas	提取钱包中的所有可提取的 GAS	需要打开钱包
create address [n=1]	创建地址 / 批量创建地址	需要打开钱包
import key \	导入私钥 / 批量导入私钥	需要打开钱包
export key [address] [path]	导出私钥	需要打开钱包
send \ \	向指定地址转账 参数分别为：资产 ID，对方地址，转账金额，手续费	需要打开钱包

以下命令可能需要详细解释一下：

?? upgrade wallet <path>

升级旧版钱包文件

```
neo>upgrade wallet cli.db3
Wallet file upgrade complete. Old file has been auto-saved at: cli.old.db3
```

?? rebuild index

重建钱包索引。为什么要重建钱包索引，重建钱包索引有什么用？

钱包中有一个字段，记录了当前钱包同步的区块高度，每新增加一个区块，钱包客户端就会同步区块，将钱包中的资产和交易更新。假设当前记录的区块高度为 100，然后你执行了 `import key` 命令导入了私钥，这时钱包仍然是从区块高度为 100 开始计算你的资产。如果导入的地址在区块高度小于 100 的时候有一些交易，这些交易和对应的资产将不会体现在钱包中，所以要重建钱包索引，强制让钱包从区块高度为 0 开始计算你的资产。

假如由于种种原因，钱包中的某笔交易未确认，这时资产已经从钱包中扣除，但并未经过整个区块链网络的确认。如果想删掉这笔未确认的交易使钱包中的资产正常显示也需要重建钱包索引。

新创建的钱包不用重建钱包索引，只有要导入私钥或者钱包中资产显示异常时才需要重建钱包索引。

?? show utxo [id|alias]

列出钱包中指定资产的 UTXO，示例如入输出如下所示：

```
neo>show utxo neo
8674c38082e59455cf35cee94a5a1f39f73b617b3093859aa199c756f7900f1f:2
total: 1 UTXOs
neo>show utxo gas
8674c38082e59455cf35cee94a5a1f39f73b617b3093859aa199c756f7900f1f:1
total: 1 UTXOs
neo>show utxo 025d82f7b00a9ff1cfe709abe3c4741a105d067178e645bc3ebad9b
c79af47d4
8674c38082e59455cf35cee94a5a1f39f73b617b3093859aa199c756f7900f1f:0
total: 1 UTXOs
```

?? show gas

列出钱包中的所有可提取及不可提取的 GAS，输出结果如下：

```
unavailable: 133.024
available: 10.123
```

其中 unavailable 表示不可提取的 GAS，available 表示可提取（待提取）的 GAS。

- 注：这里不包含已提取的 GAS，查看已提取的 GAS 请用 list asset 命令。

什么是可提取的 GAS，什么是不可提取的 GAS？

每一个 NEO 都有两种状态：unspent 和 spent。每一个未提取的 GAS 也有两种状态，available 和 unavailable。一个 NEO 的生命周期以转入地址起始，转出地址截止，转入时状态变为 unspent，转出时状态变为 spent。当 NEO 处于 unspent 状态时，所产生的 Gas 为 unavailable 状态，即不可提取。当 NEO 处于 spent 状态时，期间所产生的 GAS 变为 available，用户可以提取。

如何将钱包中的所有 unavailable GAS 转为 available GAS 呢，很简单，将钱包中的所有 NEO 转到钱包中的任意一个地址即可。

?? claim gas

提取钱包中的所有可提取的 GAS，该过程是通过一个特殊的交易 Claim Transaction 完成的，输入命令后，客户端会输出 Claim Transaction 的 ID（提取 GAS 这笔交易的 ID）。

执行完 claim gas 命令后，再执行 list asset 会显示 GAS 有增加。

?? create address [n=1]

可以输入 create address 来创建一个地址，也可以输入 create address 100 来批量创建 100 个地址，批量创建的地址会自动导出到 address.txt 文件中。

?? export key [address] [path]

你可以指定导出哪个地址对应的私钥，也可以指定导出至指定的文件中，例如下面的命令都是合法的。该命令需要验证钱包密码。

export key

export key AeSHyuirTxbfZbFik6SiBW2BEj7GK3N62b

export key key.txt

export key AeSHyuirTxbfZbFik6SiBW2BEj7GK3N62b key.txt

?? import key <wif|path>

导入私钥，你可以指定导入一个私钥，或者是导入一个文件中的多个私钥，例如下面的命令都是合法的。

```
import key L4zRFphDJpLzXZzYrYKvUoz1LkhZprS5pTYywFqTJT2EcmWPPpPH
```

```
import key key.txt
```

如果是指定文件的话，文件里的私钥格式请参考 `export key key.txt` 的输出。

```
?? send <id|alias> <address> <value>|all [fee=0]
```

转账，一共有 4 个参数，第一个参数是资产 ID，第二个参数是收款地址，第三个参数是转账金额（当输入 `all` 即为钱包中该资产的全部数量），第四个参数是手续费（这个参数可不填，默认为 0）。该命令需要验证钱包密码。假如我想转 100 NEO 转到这个地址

“AeSHyuirTXbfZbFik6SiBW2BEj7GK3N62b”，我需要输入以下命令。

```
send  
c56f33fc6ecfcd0c225c4ab356fee59390af8560be0e930faebe74a6daff7c9b  
AeSHyuirTXbfZbFik6SiBW2BEj7GK3N62b 100
```

因为第二个参数除了资产 ID，还可以填写资产缩写，如 `neo`，`gas`，所以上命令可以写成：

```
send neo AeSHyuirTXbfZbFik6SiBW2BEj7GK3N62b 100
```

不知道资产 ID 怎么办？请输入 `list asset` 命令查看钱包中的所有资产。

查看节点信息

命令	功能说明
<code>show state</code>	显示当前区块链同步状态
<code>show node</code>	显示当前已连接的节点地址和端口
<code>show pool</code>	显示内存池中的交易（这些交易处于零确认的状态）
<code>export blocks</code> <code>[path=chain.acc]</code>	导出全部区块数据，导出的结果可以用作离线同步

高级指令

命令	功能说明
<code>start consensus</code>	启动共识

启动共识的前提是该钱包有共识的权限，在 NEO 主网上可以通过投票选举获得共识的权限，如果自己部署的私有链可以在 protocol.json 中设置共识节点的公钥，详情可参考 [私链搭建](#)。

2、如何用 C# 编写 NEO 智能合约

如何用 C# 编写 NEO 智能合约

目前 NEO 智能合约推荐使用 C# 语言来开发（此外还支持 Java、Kotlin、Python 等语言开发，未来还将支持 Go、C/C++、JavaScript 等更多的编程语言）

此部分包含简短的教程，可指导你配置 NEO 智能合约的 C# 开发环境，并使你了解如何创建智能合约项目，以及如何编译。

开发工具

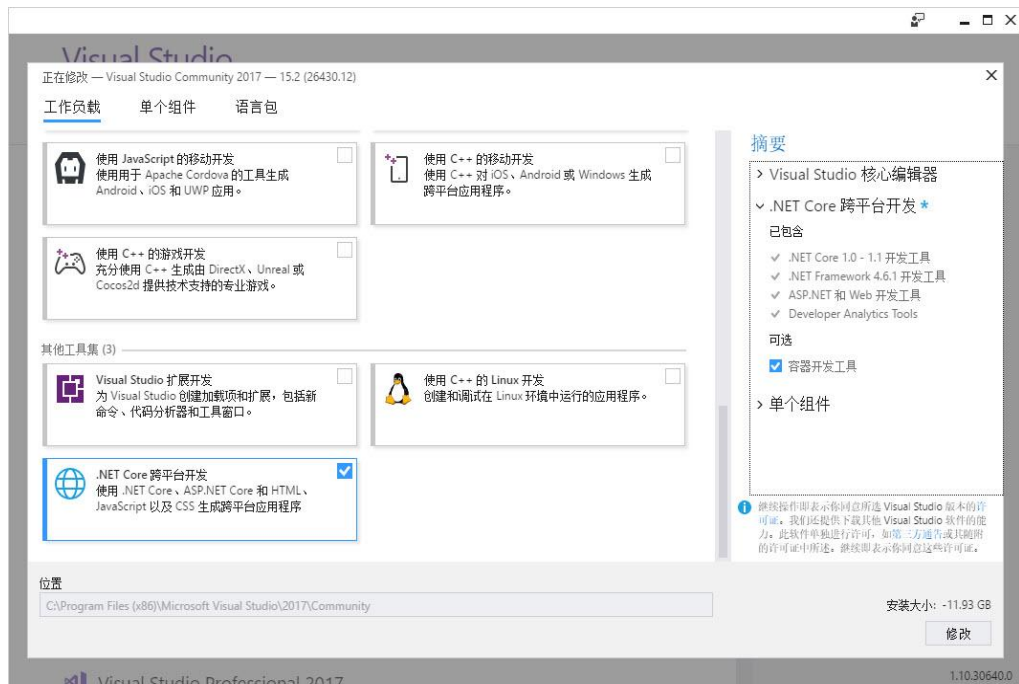
1. Visual Studio 2017

如果你的计算机中已经安装过 Visual Studio 2017，并且在安装时勾选了 .NET Core 跨平台开发 可跳过本小节。如果你电脑中安装的是 Visual Studio 2015，则无法进行下一步，请升级 Visual Studio。

下载及安装方法：

[Visual Studio 下载地址](#)

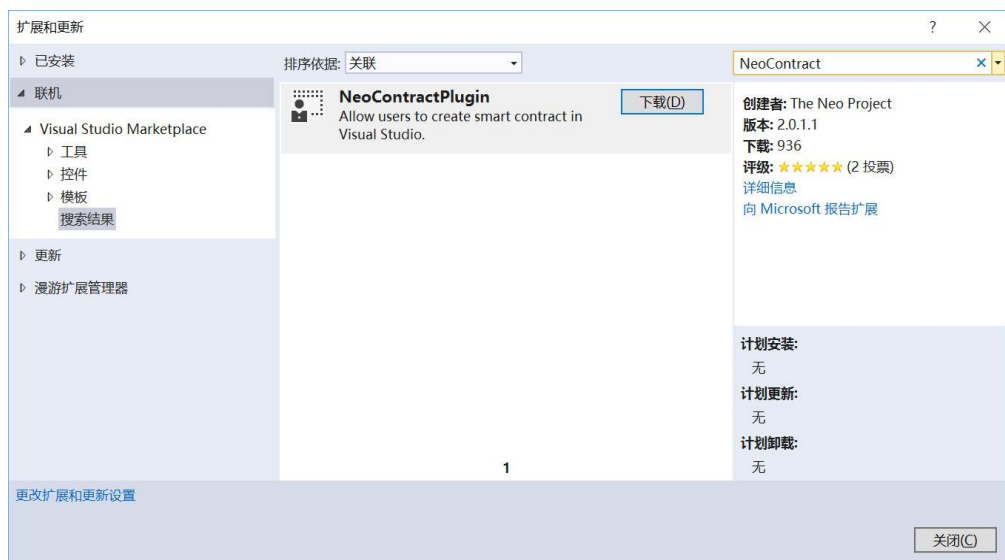
安装过程很简单，直接按照提示一步一步操作即可，需要注意的是在安装时需要勾选 .NET Core 跨平台开发，安装大概需要十几分钟或几十分钟。



2. NeoContractPlugin 插件

安装方法:

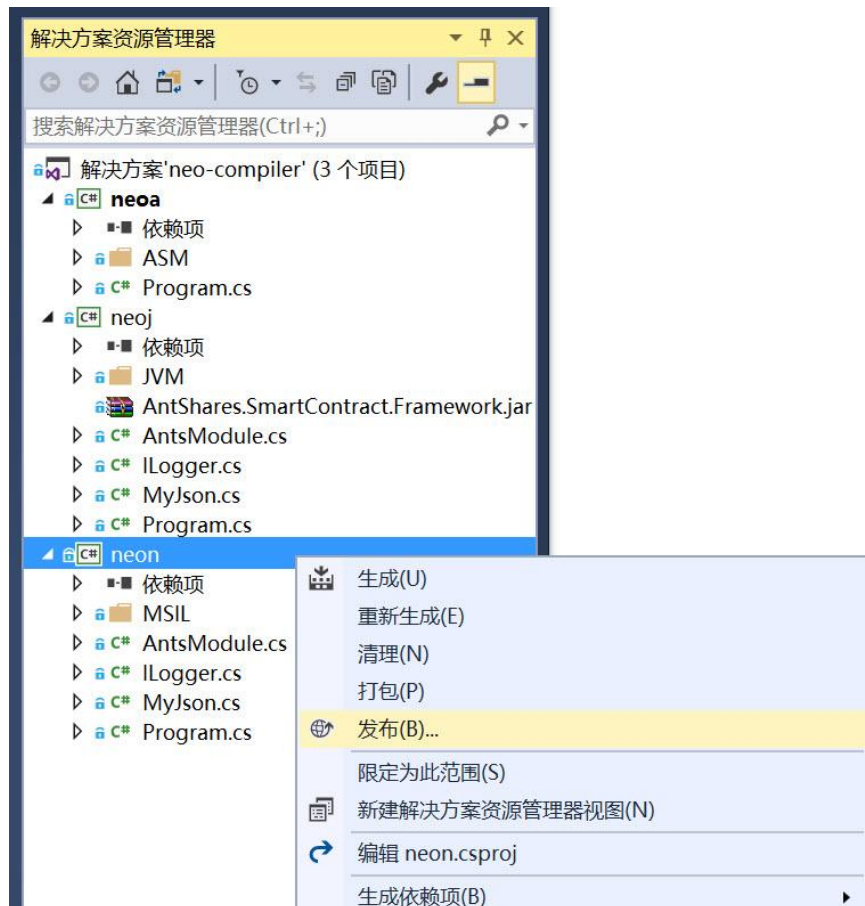
打开 Visual Studio 2017, 打开 工具, 扩展和更新, 在左侧点击 联机, 搜索 Neo, 安装 NeoContractPlugin 插件 (该过程需要联网)



3. neo-compiler

安装和配置方法:

在 Github 上下载 [neo-compiler](#) 项目, 用 Visual Studio 2017 打开该解决方案, 发布其中的 neon 项目, 如图



发布

将应用发布到 Azure 或另一台主机。 [了解更多](#)

FolderProfile

发布(P)

[新建配置文件...](#)[操作](#)

目标位置	bin\Release\PublishOut...	配置...
配置	Release	
目标框架	netcoreapp2.0	
目标运行时	win10-x64	

配置文件设置

配置文件名称: FolderProfile

配置: Release | Any CPU

目标框架: netcoreapp2.0

部署模式: 独立

目标运行时: win10-x64

目标位置: bin\Release\PublishOutput

保存

取消

neo 项目默认的发布平台为 win10-x64，如果你不是 Windows10 系统，需要修改发布平台，用文本编译器打开 neon.csproj 文件，将 `<RuntimeIdentifiers>win10-x64</RuntimeIdentifiers>` 更改为目标平台，如 `<RuntimeIdentifiers>win7-64</RuntimeIdentifiers>`，然后用

VS 重新发布该项目即可。详细 RID 可以参考 [.NET Core Runtime Identifier \(RID\) catalog](#)

> [!Note] > > 常见问题: > > [发布 neon 时提示 NuGet 错误](#) > > [无法复制 neon.dll](#)

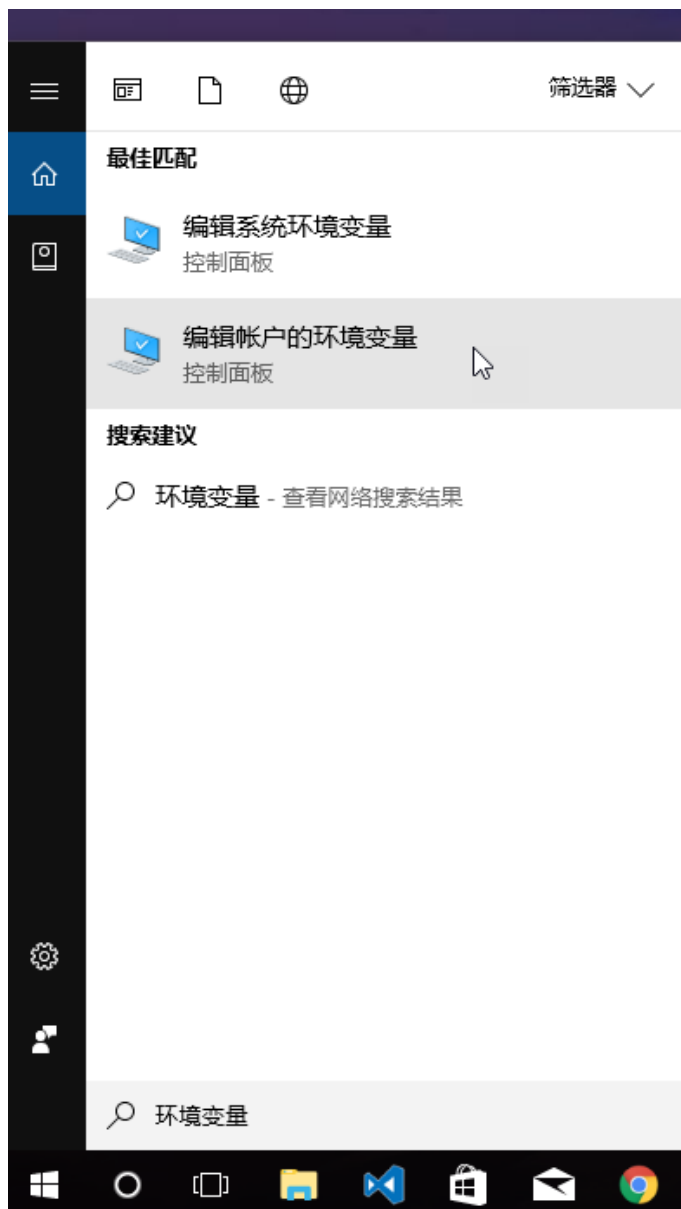
发布成功后，会在 bin\Release\PublishOutput 目录下生成 neon.exe 文件

然后需要添加 path，让任何位置都能访问这个 exe 程序

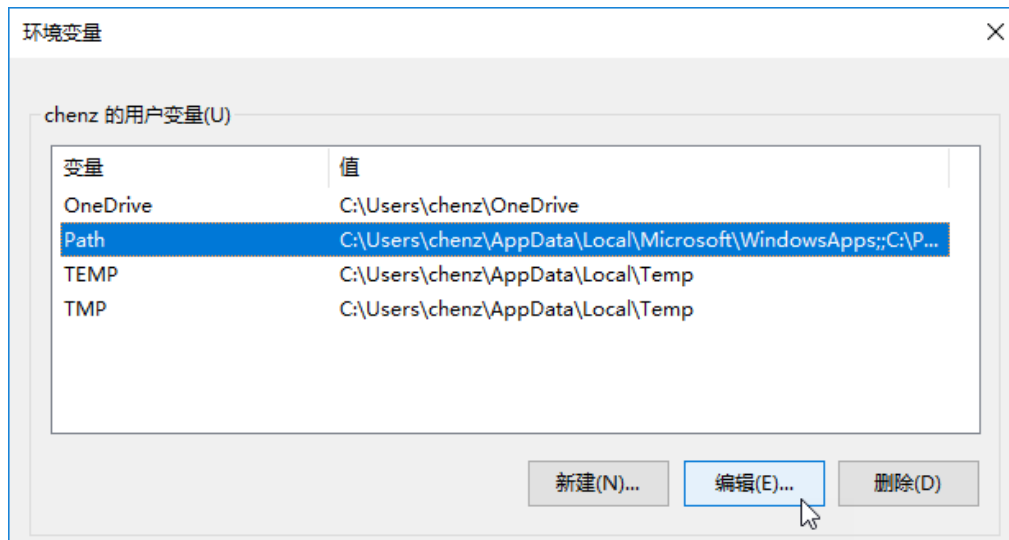
添加 path 方法：

Windows10：按 Windows + S 键，输入“环境变量”，选择“编辑账户的环境变量” 回车

Windows7 SP1 - Windows8.1 系统：右击 计算机，属性，高级系统设置，环境变量

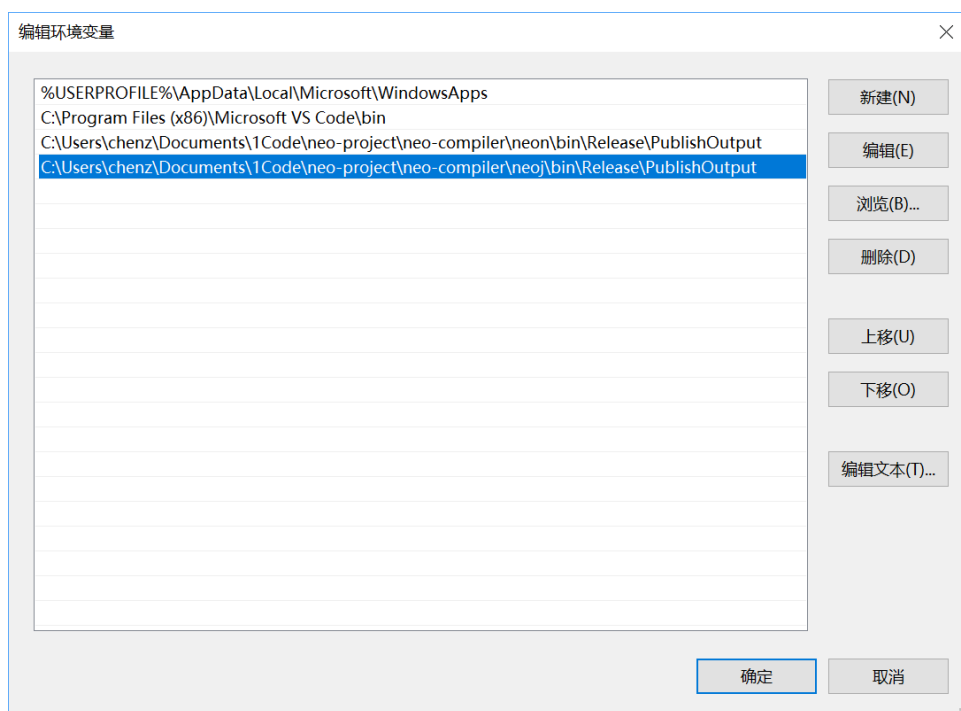


然后选择 Path, 点击 编辑



在弹出来的窗口中点击 新建 输入 neon.exe 所在的文件夹目录，点击确定，确定。

> [!Note] > 截图中为示例的路径，真正添加的要是你自己电脑上的 neon.exe 所在的文件夹目录。>> 在环境变量中不要添加 “..... neon.exe” 字样的路径，要填写 neon.exe 所在的文件夹目录 而非 neon.exe 本身的路径 >



添加完 path 后，运行 CMD 或者 PowerShell 测试一下（如果添加 path 前就已经启动了 CMD 则要关掉重启），输入 neon 后，没有报错，输出如图所示的版本号的提示信息即表示环境变量配置成功

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

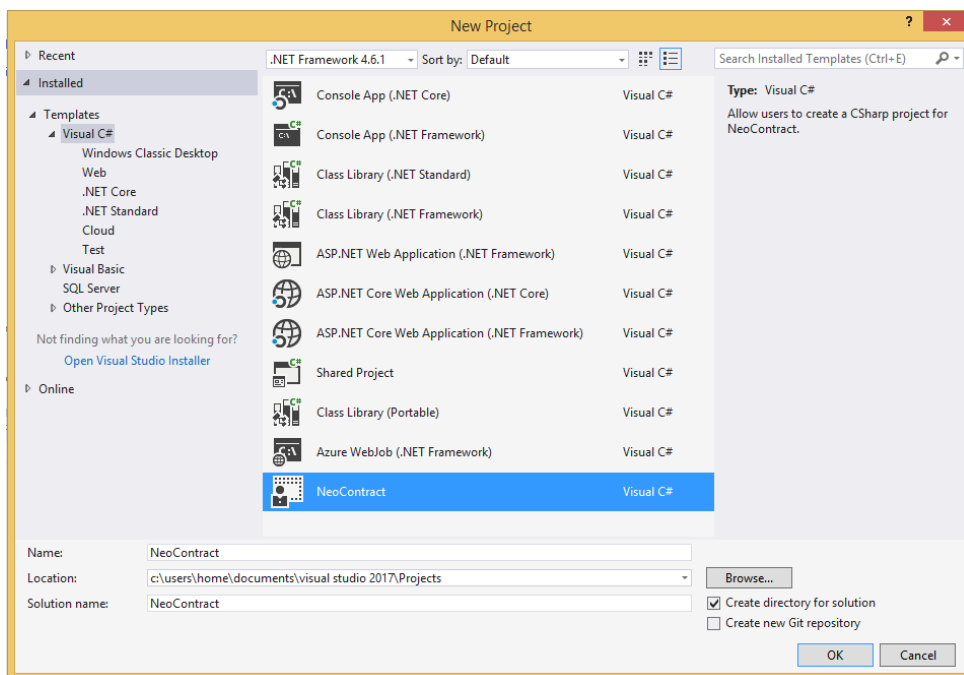
C:\Users\Home>neon
AntShars.Compiler.MSIL console app v2.0.1.0
need one param for DLL filename.

C:\Users\Home>
```

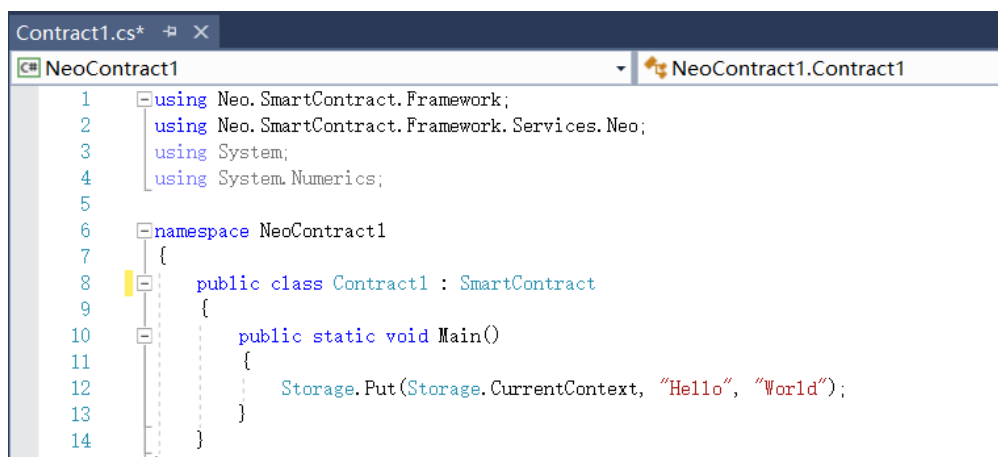
> [!Note] > > 常见问题: > > [缺少 api-ms-win-core-console-l2-1-0.dll 文件](#)

创建项目

以上四步安装配置成功后，即可在 Visual Studio 2017 中创建 NEO 智能合约项目（.NET Framework 版本任意），如图



创建项目好后，会自动生成一个 c# 文件，默认类继承于 SmartContract，如图



```
1 using Neo.SmartContract.Framework;
2 using Neo.SmartContract.Framework.Services.Neo;
3 using System;
4 using System.Numerics;
5
6 namespace NeoContract1
7 {
8     public class Contract1 : SmartContract
9     {
10         public static void Main()
11         {
12             Storage.Put(Storage.CurrentContext, "Hello", "World");
13         }
14     }
15 }
```

> [!Note] > > 常见问题: > > [新建的 NeoContract 项目找不到 Neo 命名空间](#)

编译项目

点击菜单栏上的 生成，生成解决方案（快捷键 Ctrl + Shift + B）开始编译程序。

编译成功后你会在该项目的 bin/Debug 目录下看到生成的 项目名.avm 文件，该文件即是生成的 NEO 智能合约文件。

> [!Note] > > 常见问题: > > [编译 NeoContract 时卡在 Start NeoContract converter 这一步](#)

现在，你已经完成了 NEO 智能合约开发环境的配置，更多智能合约编写方法请参考 [NEO 智能合约教程](#)

3、智能合约参数和返回值

智能合约参数和返回值

在智能合约发布或调用时，需要指定智能合约的参数，智能合约参数是 byte 类型，定义如下。

```
```c# /// /// 表示智能合约的参数类型 /// public enum
ContractParameterType : byte { /// /// 签名 /// Signature = 0,
Boolean = 1, /// /// 整数 /// Integer = 2, /// /// 160 位散列值
/// Hash160 = 3, /// /// 256 位散列值 /// Hash256 = 4, /// ///
字节数组 /// ByteArray = 5, PublicKey = 6, String = 7,
```

```
/// <summary>
```

```

 /// 对象数组
 /// </summary>
 Array = 10,
 InteropInterface = 0xf0,
 Void = 0xff
}

```

如智能合约定义为：

```

```c#
public class Lock : SmartContract
{
    public static bool Main(int a, bool b, byte[] pubkey, byte[] signature)
    {
        // 略……
    }
}

```

通过上表可查到，int 为 2，bool 为 1，公钥字节数组为 6，签名字节数组为 0。

在 NEO-GUI 客户端发布智能合约填写参数时，每个参数用两位 16 进制字符表示，所以上面的智能合约的形参列表表示为：02010600，返回值为：01。

4、系统使用费

系统使用费

交易费用

目前转账交易没有手续费。

智能合约费用

智能合约的手续费结构已经在下表列出。

每个智能合约在每次执行过程中有 10 GAS 的免费额度，无论是开发者部署还是用户调用，因此，单次执行费用在 10 GAS 以下的智能合约是不需要支付手续费的。当单次执行费用超过 10 GAS，会减免 10 GAS 的手续费。

所有支付的智能合约手续费将作为系统手续费，并按比例重新分配给所有 NEO 的持有人。

智能合约中系统调用的手续费：

SysCall	手续费 [Gas]
Runtime.CheckWitness	0.2
Blockchain.GetHeader	0.1
Blockchain.GetBlock	0.2
Blockchain.GetTransaction	0.1
Blockchain.GetAccount	0.1
Blockchain.GetValidators	0.2
Blockchain.GetAsset	0.1
Blockchain.GetContract	0.1
Transaction.GetReferences	0.2
Account.SetVotes	1
Validator.Register	1000
Asset.Create（系统资产）	5000
Asset.Renew（系统资产）	5000
Contract.Create*	100~1000
Contract.Migrate*	100~1000
Storage.Get	0.1
Storage.Put [per KB]	1
Storage.Delete	0.1
其它（每行 OpCode）	0.001

关于表格中 API 的含义，请参见 [NEO 命名空间](#)。

* 创建智能合约与迁移智能合约目前是根据合约所需功能进行收费。其中基础的费用为 100GAS，需要存储区 +400GAS，需要动态调用 +500GAS。

指令费用

Instruction	手续费 [Gas]
OpCode.PUSH16 [or less]	0
OpCode.NOP	0
OpCode.APPCALL	0.01
OpCode.TAILCALL	0.01
OpCode.SHA1	0.01
OpCode.SHA256	0.01
OpCode.HASH160	0.02

Instruction	手续费 [Gas]
OpCode. HASH256	0.02
OpCode. CHECKSIG	0.1
OpCode. CHECKMULTISIG (每个签名)	0.1
其它 (每行 OpCode)	0.001

1、项目中使用的智能合约

1.1、智能合约概述

合约说明

角斗士项目中总共使用三个合约：sgas，NFT 和 Auction，其中 Sgas 合约为 NEL 开发的通用合约，NFT 和 Auction 为我们自己开发。 NFT 合约管理角斗士资源，以及克隆操作； Auction 合约管理角斗士的拍卖，出租，购买和手续费扣取。 本项目所用合约均使用 C#开发。

1. Sgas.cs 作用：用于 gas 和 sgas 之间 1: 1 兑换。 由于合约里直接操作 gas 比较困难，所以需要玩家先将 gas 通过 sgas 合约转换为 NEP5 资产，合约里操控的是 sgas 资产，这样合约写起来会比较方便。
2. NFT.cs 作用：管理角斗士资源，以及克隆操作。
3. Auction.cs 作用：管理角斗士的拍卖，出租，购买和手续费扣取。

1.2、合约常见问题和注意事项

常见问题：

1. 如何获取我拥有的角斗士资产？ 这个应该是由 tokensOfOwner 接口提供的，但是如果要在链上统计获取这些数据，角斗士比较多时候，这个接口的手续费肯定是要超过 10GAS 的，所以该接口暂不支持。 现在我们的做法是在产生角斗士的时候，发出通知 (ApplicationLog)，后台监听到通知后，入后台数据库进行记录，在前端需要的时候由后台提供接口获取角斗士数据。
2. 如何充值到拍卖行 玩家先将 gas 通过 sgas 合约，充值进入合约转换成 sgas，再将 sgas 转账给拍卖合约所代表的钱包地址，最后将该转账的交易 txid，传给 Auction 拍卖合约，Auction 合约会去 sgas 合约里查询交易记录，看这笔记录是否转账成功以及转账金额，将该金额记录在拍卖合约的 storage 中，即充值成功。为避免重复充值，拍卖合约中会标记哪些 txid 已经处理过。

3. 玩家在拍卖场里买东西为什么要提前充值到拍卖行 因为如果不提前充值到拍卖行，买东西的时候，合约没有办法主动扣除玩家的 gas 或者 sgas。
4. 关于结构体的序列化 2.7 以后的版本支持使用 `Helper.Deserialize` 进行序列化，2.6 的版本不支持该方法，目前测试网络的版本为 2.6，主网的版本为 2.7

注意事项：

1. 关于手续费 官方文档上说：“每个智能合约在每次执行过程中有 10 GAS 的免费额度，无论是开发者部署还是用户调用，因此，单次执行费用在 10 GAS 以下的智能合约是不需要支付手续费的。当单次执行费用超过 10 GAS，会减免 10 GAS 的手续费。”。我们在写智能合约时，应该控制合约单个执行函数的复杂度，尽量将手续费控制在 10GAS 以内，这样系统减免后就可以免费调用。为了将手续费控制在 10GAS 以内，`Storage.Put` 调用的次数在一个调用内尽量不要超过 6 个；尽量避免遍历读取 `storage` 里的数据；尽量避免在循环里多次调用一个复杂的私有函数。
2. 关于数据类型 智能合约中没有 `string` 类型，使用 `Byte` 字节数组来表示 `string`，不要将 `int` 等类型转换为 `string`，不支持该操作；智能合约中数值类型只有 `BigInteger`，最终也被存储为 `ByteArray`。C# 中的 `byte`，`int`，`long` 等都会被自动编译成 `BigInteger` 类型。智能合约中不支持 C# 的浮点小数，可以使用 `BigInteger` 表示的定点小数，需要约定小数位数。C# 合约中的结构体，不支持自定义构造函数，不支持在其中自定义成员函数。
3. 其他常见错误：不要在智能合约中定义类变量，不支持；可以定义类常量。

2、Gladiator 合约接口

2.1、name

简要描述：

- 合约名称

接口名：

- `name`

参数：

参数名	类型	说明
-----	----	----

参数名	类型	说明
无		

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "00c1046e616d65675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.087",
    "stack" : {
      "0" : {
        "type" : "ByteArray",
        "value" : "4372617a79476c61646961746f72"
      },
    },
  }
}
```

返回参数说明

参数名	类型	说明
0	string	合约名称:CrazyGladiator

2.2、symbol

简要描述:

- 合约符号

接口名:

- symbol

参数:

参数名	类型	说明
无		

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "00c10673796d626f6c675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.102",
    "stack" : {
      "0" : {
        "type" : "ByteArray",
        "value" : "43474c"
      },
    },
  }
}
```

返回参数说明

参数名	类型	说明
0	string	合约符号:CGL

2.3、decimals

简要描述:

- 精度

接口名:

- decimals

参数:

参数名	类型	说明
无		

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "00c108646563696d616c73675f31fb4260c116e978efd3c
```

```

2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.111",
    "stack" : {
        "0" : {
            "type" : "ByteArray",
            "value" : ""
        },
    },
}
}
}

```

返回参数说明

参数名	类型	说明
0	number	精度

2.4、totalSupply

简要描述:

- 已经发行的角斗士总数

接口名:

- totalSupply

参数:

参数名	类型	说明
无		

返回示例

```

{
    "jsonrpc" : "2.0",
    "id" : 1,
    "result" : {
        "script" : "00c10b746f74616c537570706c79675f31fb4260c116e978efd3c2e874a56ad839eded",
        "state" : "HALT, BREAK",
        "gas_consumed" : "0.244",
        "stack" : {
            "0" : {

```

```
        "type" : "ByteArray",
        "value" : "6a27"
    },
}
}
```

返回参数说明

参数名	类型	说明
0	number	已经发行的角斗士总数

2.5、ownerOf

简要描述:

- 获取角斗士拥有者

接口名:

- ownerOf

参数:

参数名	类型	说明
tokenId	BigInteger	角斗士 ID

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "5151c1076f776e65724f66675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.414",
    "stack" : {
      "0" : {
        "type" : "ByteArray",
        "value" : "6b0b89dfc5d7f5f04ce2a1d9051865f611e7e3a0"
      },
    },
  },
}
```

```
}
}
```

返回参数说明

参数名	类型	说明
0	string	角斗士拥有者的钱包地址

2.6、tokenURI

简要描述：

- uri

接口名：

- tokenURI

参数：

参数名	类型	说明
tokenId	BigInteger	角斗士 ID

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "5151c108746f6b656e555249675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.526",
    "stack" : {
      "0" : {
        "type" : "ByteArray",
        "value" : "7572692f01"
      },
    },
  }
}
```

返回参数说明

参数名	类型	说明
0	string	uri

2.7、mintToken

简要描述：

- 发行促销角斗士

接口名：

- mintToken

参数：

参数名	类型	说明
tokenOwner	byte[]	角斗士发给谁
strength	byte	力量
power	byte	体力
agile	byte	
speed	byte	速度
skill1	byte	技能
skill2	byte	技能
skill3	byte	技能
skill4	byte	技能
skill5	byte	技能
equip1	byte	装备
equip2	byte	装备
equip3	byte	装备
equip4	byte	装备
restrictAttribute	byte	元素
character	byte	形象
part1	byte	外观属性
part2	byte	
part3	byte	
part4	byte	
part5	byte	
appear5	byte	
appear6	byte	
appear7	byte	
appear8	byte	

参数名	类型	说明
appear9	byte	
chest	byte	
bracer	byte	
shoulder	byte	
face	byte	暂未使用
lip	byte	暂未使用
nose	byte	暂未使用
eyes	byte	暂未使用
hair	byte	暂未使用
<pre>{ "jsonrpc" : "2.0", "id" : 1, "result" : "true" }</pre>		

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

2.8、giveBirth

简要描述：

- 角斗士出生

接口名：

- giveBirth

参数：

参数名	类型	说明
motherId	BigInteger	已经准备好生小孩的角斗士 id
<pre>{ "jsonrpc" : "2.0", "id" : 1, "result" : "true" }</pre>		

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

2.9、isReadyToBreed

简要描述：

- 怀孕的角斗士是否到了生小孩的时间

接口名：

- isReadyToBreed

参数：

参数名	类型	说明
tokenId	BigInteger	角斗士 ID

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "5151c10e69735265616479546f4272656564675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.606",
    "stack" : {
      "0" : {
        "type" : "ByteArray",
        "value" : ""
      },
    },
  }
}
```

返回参数说明

参数名	类型	说明
0	Bool	是否到了生小孩的时间

2.10、isPregnant

简要描述:

- 判断角斗士是否怀孕

接口名:

- isPregnant

参数:

参数名	类型	说明
tokenId	BigInteger	角斗士 ID

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "5151c10a6973507265676e616e74675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.651",
    "stack" : {
      "0" : {
        "type" : "ByteArray",
        "value" : ""
      },
    },
  },
}
```

返回参数说明

参数名	类型	说明
0	bool	判断角斗士是否怀孕

2.11、canBreedWithById

简要描述:

- 判断两个角斗士是否可以进行克隆操作

接口名:

- canBreedWithById

参数:

参数名	类型	说明
montherId	BigInteger	母亲角斗士 ID
fatherId	BigInteger	父亲角斗士 ID

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "525152c11063616e42726565645769746842794964675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "1.004",
    "stack" : {
      "0" : {
        "type" : "Integer",
        "value" : "1"
      },
    },
  },
}
```

返回参数说明

参数名	类型	说明
0	bool	是否可以克隆操作

2.12、transfer

简要描述:

- 将角斗士资产转账给其他人

接口名:

- transfer

参数:

参数名	类型	说明
-----	----	----

参数名	类型	说明
from	byte[]	转出钱包地址
to	byte[]	转入钱包地址
tokenId	BigInteger	角斗士 ID
<pre>{ "jsonrpc" : "2.0", "id" : 1, "result" : "true" }</pre>		

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

2.13、tokenData

简要描述:

- 获取角斗士信息

接口名:

- tokenData

参数:

参数名	类型	说明
tokenId	BigInteger	角斗士 ID

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "5151c109746f6b656e44617461675f31fb4260c116e978e
fd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.531",
    "stack" : {
      "0" : {
        "type" : "Array",
        "value" : {
```

65f611e7e3a0"

```
"0" : {
  "type" : "ByteArray",
  "value" : "6b0b89dfc5d7f5f04ce2ald90518
65f611e7e3a0"
},
"1" : {
  "type" : "ByteArray",
  "value" : ""
},
"2" : {
  "type" : "ByteArray",
  "value" : ""
},
"3" : {
  "type" : "Integer",
  "value" : "1"
},
"4" : {
  "type" : "ByteArray",
  "value" : ""
},
"5" : {
  "type" : "ByteArray",
  "value" : ""
},
"6" : {
  "type" : "Integer",
  "value" : "1527160861"
},
"7" : {
  "type" : "ByteArray",
  "value" : ""
},
"8" : {
  "type" : "ByteArray",
  "value" : ""
},
"9" : {
  "type" : "ByteArray",
  "value" : ""
},
"10" : {
  "type" : "ByteArray",
  "value" : "5d"
```

```
},
"11" : {
  "type" : "ByteArray",
  "value" : "40"
},
"12" : {
  "type" : "ByteArray",
  "value" : "30"
},
"13" : {
  "type" : "ByteArray",
  "value" : "23"
},
"14" : {
  "type" : "ByteArray",
  "value" : "d800"
},
"15" : {
  "type" : "Integer",
  "value" : "4"
},
"16" : {
  "type" : "ByteArray",
  "value" : "de00"
},
"17" : {
  "type" : "ByteArray",
  "value" : ""
},
"18" : {
  "type" : "ByteArray",
  "value" : ""
},
"19" : {
  "type" : "ByteArray",
  "value" : "d600"
},
"20" : {
  "type" : "ByteArray",
  "value" : "d300"
},
"21" : {
  "type" : "ByteArray",
  "value" : ""
}
```

```
},
"22" : {
  "type" : "ByteArray",
  "value" : ""
},
"23" : {
  "type" : "Integer",
  "value" : "1"
},
"24" : {
  "type" : "Integer",
  "value" : "2"
},
"25" : {
  "type" : "Integer",
  "value" : "5"
},
"26" : {
  "type" : "Integer",
  "value" : "2"
},
"27" : {
  "type" : "Integer",
  "value" : "5"
},
"28" : {
  "type" : "Integer",
  "value" : "9"
},
"29" : {
  "type" : "Integer",
  "value" : "1"
},
"30" : {
  "type" : "Integer",
  "value" : "1"
},
"31" : {
  "type" : "Integer",
  "value" : "5"
},
"32" : {
  "type" : "ByteArray",
  "value" : "553764c100"
```



```

    },
    "33" : {
        "type" : "ByteArray",
        "value" : "553764c100"
    },
    "34" : {
        "type" : "ByteArray",
        "value" : "553764c100"
    },
    "35" : {
        "type" : "ByteArray",
        "value" : "553764c100"
    },
    "36" : {
        "type" : "ByteArray",
        "value" : "553764c100"
    },
    "37" : {
        "type" : "ByteArray",
        "value" : "553764c100"
    },
    "38" : {
        "type" : "ByteArray",
        "value" : "553764c100"
    },
    "39" : {
        "type" : "ByteArray",
        "value" : "553764c100"
    },
    "40" : {
        "type" : "ByteArray",
        "value" : "553764c100"
    },
    "41" : {
        "type" : "ByteArray",
        "value" : "553764c100"
    },
    "42" : {
        "type" : "ByteArray",
        "value" : "553764c100"
    },
    }
}

```

```
}  
}
```

返回参数说明

参数	类型	描述
- 0	object	地址
- 1	object	isGestating
- 2	object	isReady
- 3	object	cooldownIndex
- 4	object	nextActionAt
- 5	object	cloneWithId
- 6	object	birthTime
- 7	object	matronId
- 8	object	sireId
- 9	object	generation
- 10	object	strength
- 11	object	power
- 12	object	agile
- 13	object	speed
- 14	object	skill1
- 15	object	skill2
- 16	object	skill3
- 17	object	skill4
- 18	object	skill5
- 19	object	equip1
- 20	object	equip2
- 21	object	equip3
- 22	object	equip4
- 23	object	restrictAttribute
- 24	object	character
- 25	object	part1
- 26	object	part2
- 27	object	part3
- 28	object	part4
- 29	object	part5
- 30	object	appear1
- 31	object	appear2
- 32	object	appear3

参数	类型	描述
- 33	object	appear4
- 34	object	appear5
- 35	object	chest
- 36	object	bracer
- 37	object	暂未使用
- 38	object	暂未使用
- 39	object	暂未使用
- 40	object	暂未使用
- 41	object	暂未使用
- 42	object	暂未使用

2.14、getAuctionAddr

简要描述：

- 获取拍卖行地址

接口名：

- getAuctionAddr

参数：

参数名	类型	说明
无		

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "00c10e67657441756374696f6e41646472675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.499",
    "stack" : {
      "0" : {
        "type" : "ByteArray",
        "value" : "44bc0ee064759cd08860dc73a8dbd4ace4185c60"
      }
    }
  },
}
```

```
    }  
  }  
}
```

返回参数说明

参数名	类型	说明
0	byte[]	拍卖行地址

2.15、setAuctionAddr

简要描述:

- 设置拍卖行地址

接口名:

- setAuctionAddr

参数:

参数名	类型	说明
auctionAddr	byte[]	拍卖行地址
<pre>{ "jsonrpc" : "2.0", "id" : 1, "result" : "true" }</pre>		

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

2.16、getAttrConfig

简要描述:

- 获取技能武器属性配置参数

接口名:

- getAttrConfig

参数:

参数名	类型	说明
无		

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "00c10d67657441747472436f6e666967675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.6",
    "stack" : {
      "0" : {
        "type" : "Array",
        "value" : {
          "0" : {
            "type" : "Integer",
            "value" : "1"
          },
          "1" : {
            "type" : "ByteArray",
            "value" : "15"
          },
          "2" : {
            "type" : "ByteArray",
            "value" : "c900"
          },
          "3" : {
            "type" : "ByteArray",
            "value" : "ea00"
          },
          "4" : {
            "type" : "Integer",
            "value" : "1"
          },
          "5" : {
            "type" : "ByteArray",
            "value" : "1f"
          },
          "6" : {
            "type" : "ByteArray",
            "value" : "c900"
          }
        }
      }
    }
  }
}
```

```

    },
    "7" : {
        "type" : "ByteArray",
        "value" : "e000"
    },
    "8" : {
        "type" : "Integer",
        "value" : "10"
    },
    "9" : {
        "type" : "Integer",
        "value" : "9"
    },
    "10" : {
        "type" : "Integer",
        "value" : "3"
    },
    "11" : {
        "type" : "Integer",
        "value" : "10"
    },
    "12" : {
        "type" : "Integer",
        "value" : "10"
    },
    "13" : {
        "type" : "Integer",
        "value" : "3"
    },
    "14" : {
        "type" : "Integer",
        "value" : "9"
    },
    "15" : {
        "type" : "Integer",
        "value" : "9"
    },
    "16" : {
        "type" : "Integer",
        "value" : "9"
    },
}
}
}

```

```
}  
}
```

返回参数说明

参数	类型	描述
- 0	BigInteger	normalSkillIdMax
- 1	BigInteger	normalSkillIdMax
- 2	BigInteger	rareSkillIdMin
- 3	BigInteger	rareSkillIdMax
- 4	BigInteger	normalEquipIdMin
- 5	BigInteger	normalEquipIdMax
- 6	BigInteger	rareEquipIdMin
- 7	BigInteger	rareEquipIdMax
- 8	BigInteger	atr1Max
- 9	BigInteger	atr2Max
- 10	BigInteger	atr3Max
- 11	BigInteger	atr4Max
- 12	BigInteger	atr5Max
- 13	BigInteger	atr6Max
- 14	BigInteger	atr7Max
- 15	BigInteger	atr8Max
- 16	BigInteger	atr9Max

2.17、setAttrConfig

简要描述:

- 设置技能武器配置参数

接口名:

- setAttrConfig

参数:

参数名	类型	说明
normalSkillIdMax	BigInteger	普通技能 id 最小值
normalSkillIdMax	BigInteger	普通技能 id 最大值
rareSkillIdMin	BigInteger	rareSkillIdMin
rareSkillIdMax	BigInteger	rareSkillIdMax

参数名	类型	说明
normalEquipIdMin	BigInteger	normalEquipIdMin
normalEquipIdMax	BigInteger	normalEquipIdMax
rareEquipIdMin	BigInteger	rareEquipIdMin
rareEquipIdMax	BigInteger	rareEquipIdMax
atr1Max	BigInteger	atr1Max
atr2Max	BigInteger	atr2Max
atr3Max	BigInteger	atr3Max
atr4Max	BigInteger	atr4Max
atr5Max	BigInteger	atr5Max
atr6Max	BigInteger	atr6Max
atr7Max	BigInteger	atr7Max
atr8Max	BigInteger	atr8Max
atr9Max	BigInteger	atr9Max
<pre>{ "jsonrpc" : "2.0", "id" : 1, "result" : "true" }</pre>		

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

2.18、getTXInfo

简要描述:

- 获取交易信息

接口名:

- getTXInfo

参数:

参数名	类型	说明
txid	byte[]	交易 id

返回示例


```

{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "20f6912fdedaf997cb8d7e62c907110cab3122d49a6f561
bb7db50ce6ff8b9b94a51c1096765745458496e666f675f31fb4260c116e978efd3c2
e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.52",
    "stack" : {
      "0" : {
        "type" : "Array",
        "value" : {
          "0" : {
            "type" : "ByteArray",
            "value" : "6b0b89dfc5d7f5f04ce2a1d90518
65f611e7e3a0"
          },
          "1" : {
            "type" : "ByteArray",
            "value" : "e14d105e7d750e38d78cefacb236
2a2ba5e67944"
          },
          "2" : {
            "type" : "Integer",
            "value" : "2"
          }
        }
      }
    }
  }
}

```

返回参数说明

参数名	类型	说明
0	byte[]	from
1	byte[]	to
2	byte[]	value

3、Auction 合约接口

3.1、name

简要描述:

- 合约名称

接口名:

- name

参数:

参数名	类型	说明
无		

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "00c1046e616d65675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.087",
    "stack" : {
      "0" : {
        "type" : "ByteArray",
        "value" : "4372617a79476c61646961746f724175637469666e"
      },
    },
  }
}
```

返回参数说明

参数名	类型	说明
0	string	合约名称:CrazyGladiatorAuction

3.2、totalExchangeSgas

简要描述:

- 不包含收取的手续费在内, 所有用户存在拍卖行中的代币

接口名:

- totalExchangeSgas

参数:

参数名	类型	说明
无		

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "00c10673796d626f6c675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.102",
    "stack" : {
      "0" : {
        "type" : "ByteArray",
        "value" : "1000000000"
      },
    },
  }
}
```

返回参数说明

参数名	类型	说明
0	number	不包含收取的手续费在内, 所有用户存在拍卖行中的代币

3.3、balanceOf

简要描述:

- 用户在拍卖所存储的代币

接口名:

- balanceOf

参数:

参数名	类型	说明
-----	----	----

参数名	类型	说明
address	byte[]	用户在拍卖所存储的代币

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "00c10673796d626f6c675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.102",
    "stack" : {
      "0" : {
        "type" : "ByteArray",
        "value" : "10000"
      },
    },
  }
}
```

返回参数说明

参数名	类型	说明
0	number	用户在拍卖所存储的代币

3.4、rechargeToken

简要描述:

- 使用 txid 充值

接口名:

- rechargeToken

参数:

参数名	类型	说明
owner	byte[]	充值到该地址
txid	byte[]	交易 id

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "true"
}
```

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

3.5、drawToken

简要描述：

- 提币

接口名：

- drawToken

参数：

参数名	类型	说明
owner	byte[]	充值到该地址
txid	byte[]	交易 id

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "true"
}
```

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

3.6、createSaleAuction

简要描述：

- 创建拍卖

接口名:

- createSaleAuction

参数:

参数名	类型	说明
tokenOwner	byte[]	拍卖人的钱包地址
tokenId	BigInteger	要拍卖的角斗士 ID
beginPrice	BigInteger	开始价格
endPrice	BigInteger	结束价格
duration	BigInteger	持续时间

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "true"
}
```

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

3.7、createCloneAuction

简要描述:

- 克隆拍卖创建

接口名:

- createCloneAuction

参数:

参数名	类型	说明
tokenOwner	byte[]	克隆拍卖人的钱包地址
tokenId	BigInteger	要克隆拍卖的角斗士 ID
beginPrice	BigInteger	开始价格

参数名	类型	说明
endPrice	BigInteger	结束价格
duration	BigInteger	持续时间

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "true"
}
```

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

3.8、buyOnAuction

简要描述：

- 从拍卖场购买, 将钱划入合约名下，将物品给买家

接口名：

- buyOnAuction

参数：

参数名	类型	说明
sender	byte[]	购买者的钱包地址
tokenId	BigInteger	需要购买的角斗士

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "true"
}
```

返回参数说明

参数名	类型	说明
-----	----	----

参数名	类型	说明
result	bool	true 发送成功

3.9、cloneOnAuction

简要描述：

- 购买拍卖克隆

接口名：

- cloneOnAuction

参数：

参数名	类型	说明
sender	byte[]	购买者的钱包地址
motherId	BigInteger	母亲角斗士 ID
fatherId	BigInteger	父亲角斗士 ID

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "true"
}
```

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

3.10、breedWithMy

简要描述：

- 和自己的角斗士进行克隆

接口名：

- breedWithMy

参数：

参数名	类型	说明
sender	byte[]	钱包地址
motherId	BigInteger	母亲角斗士 ID
fatherId	BigInteger	父亲角斗士 ID

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "true"
}
```

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

3.11、cancelAuction

简要描述：

- 取消拍卖

接口名：

- cancelAuction

参数：

参数名	类型	说明
sender	byte[]	钱包地址
fatherId	BigInteger	父亲角斗士 ID

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "true"
}
```

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

3.12、getAuctionById

简要描述：

- 获取拍卖信息

接口名：

- getAuctionById

参数：

参数名	类型	说明
tokenId	BigInteger	角斗士 id

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "00c10673796d626f6c675f31fb4260c116e978efd3c2e874a56ad839eded",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.102",
    "stack" : {
      "0" : {
        "type" : "ByteArray",
        "value" : "4e4654"
      },
    },
  }
}
```

返回参数说明

参数名	类型	说明
0	number	精度

3.13、createGen0Auction

简要描述:

- 发布 0 代角斗士到拍卖场

接口名:

- createGen0Auction

参数:

参数名	类型	说明
strength	byte	力量
power	byte	体力
agile	byte	
speed	byte	速度
skill1	byte	技能
skill2	byte	技能
skill3	byte	技能
skill4	byte	技能
skill5	byte	技能
equip1	byte	装备
equip2	byte	装备
equip3	byte	装备
equip4	byte	装备
restrictAttribute	byte	元素
character	byte	形象
part1	byte	外观属性
part2	byte	
part3	byte	
part4	byte	
part5	byte	
appear5	byte	
appear6	byte	
appear7	byte	
appear8	byte	
appear9	byte	
chest	byte	
bracer	byte	
shoulder	byte	
face	byte	暂未使用

参数名	类型	说明
lip	byte	暂未使用
nose	byte	暂未使用
eyes	byte	暂未使用
hair	byte	暂未使用

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "true"
}
```

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

3.14、drawToContractOwner

简要描述：

- 将收入提款到合约所有者

接口名：

- drawToContractOwner

参数：

参数名	类型	说明
count	BigInteger	提款金额

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : "true"
}
```

返回参数说明

参数名	类型	说明
result	bool	true 发送成功

3.15、getAuctionRecord

简要描述:

- 获取拍卖成交记录

接口名:

- getAuctionRecord

参数:

参数名	类型	说明
tokenId	BigInteger	角斗士 id

返回示例

```
{
  "jsonrpc" : "2.0",
  "id" : 1,
  "result" : {
    "script" : "02580251c10e67657441756374696f6e427949646744bc0
ee064759cd08860dc73a8dbd4ace4185c60",
    "state" : "HALT, BREAK",
    "gas_consumed" : "0.473",
    "stack" : {
      "0" : {
        "type" : "Array",
        "value" : {
          "0" : {
            "type" : "ByteArray",
            "value" : "e14d105e7d750e38d78cefacb236
2a2ba5e67944"
          },
          "1" : {
            "type" : "ByteArray",
            "value" : ""
          },
          "2" : {
            "type" : "Integer",
            "value" : "1527180680"
          }
        }
      }
    }
  }
}
```

```
    },
    "3" : {
        "type" : "ByteArray",
        "value" : "002d3101"
    },
    "4" : {
        "type" : "ByteArray",
        "value" : "40420f"
    },
    "5" : {
        "type" : "ByteArray",
        "value" : "805101"
    },
    },
    },
    },
    },
    }
```

返回参数说明

参数名	类型	说明
0	byte[]	拍卖者的钱包地址
1	int	拍卖类型：0 拍卖 1 克隆拍卖
2	uint	拍卖时间
3	BigInteger	拍卖开始价格
4	BigInteger	拍卖结束价格
5	BigInteger	持续时间