目录

[Goopal3.0智能合约API 3](#_Toc458520008)

[核心功能API： 3](#_Toc458520009)

[1. 编译合约 3](#_Toc458520010)

[2. 注册合约(链上) 3](#_Toc458520011)

[3. 注册合约(仅本地) 4](#_Toc458520012)

[4. 调用合约(链上) 4](#_Toc458520013)

[5. 调用合约(仅本地) 5](#_Toc458520014)

[6. 升级合约 5](#_Toc458520015)

[7. 销毁合约 6](#_Toc458520016)

[8. 转账到合约(链上) 7](#_Toc458520017)

[9. 转账到合约(仅本地) 9](#_Toc458520018)

[10. 打开/关闭本地解释器验证开关（仅对普通节点有效） 9](#_Toc458520019)

[接口功能API： 10](#_Toc458520020)

[11. 根据合约地址/名字获取合约信息 10](#_Toc458520021)

[12. 根据合约地址/名字获取合约balance 11](#_Toc458520022)

[13. 导出合约到文件 12](#_Toc458520023)

[14. 获取是否打开本地解释器验证（仅对普通节点有效） 12](#_Toc458520024)

[Sandbox功能API 13](#_Toc458520025)

[15. 打开沙盒 13](#_Toc458520026)

[16. 关闭沙盒 13](#_Toc458520027)

[17. 沙盒中注册合约 14](#_Toc458520028)

[18. 沙盒中调用合约 14](#_Toc458520029)

[19. 沙盒中升级合约 15](#_Toc458520030)

[20. 沙盒中销毁合约 16](#_Toc458520031)

[21. 沙盒中转账到合约 17](#_Toc458520032)

[22. 沙盒中获取合约信息 18](#_Toc458520033)

[23. 沙盒中获取合约balance 18](#_Toc458520034)

[Event功能API： 19](#_Toc458520035)

[24. 编译本地脚本 19](#_Toc458520036)

[25. 列出打开钱包中的所有本地脚本 19](#_Toc458520037)

[26. 查询打开钱包中的指定的本地脚本 20](#_Toc458520038)

[27. 在钱包中添加新的本地脚本 20](#_Toc458520039)

[28. 从钱包中删除本地脚本 21](#_Toc458520040)

[29. 禁用钱包中的一个本地脚本 21](#_Toc458520041)

[30. 启用钱包中的一个本地脚本 22](#_Toc458520042)

[31. 查询与合约中的Event相关的本地脚本 22](#_Toc458520043)

[32. 向合约中的Event添加一个相关的本地脚本 23](#_Toc458520044)

[33. 从合约中的Event删除一个相关的本地脚本 23](#_Toc458520045)

[34. 导出本地脚本库以及与Event的关系库 24](#_Toc458520046)

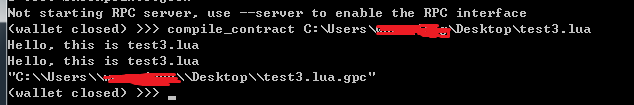
[35. 导入本地脚本库以及与Event的关系库 25](#_Toc458520047)

# Goopal3.0智能合约API

## 核心功能API：

### 编译合约

* **命令行命令**：compile\_contract
* **参数**：待编译的合约文件（文件为lua源码文件，文件后缀为.lua）
* **返回值**：生成的合约字节码文件，文件后缀为.gpc
* **示例**：



备注：中间可能会有合约中的一些相关输出.

### 注册合约(链上)

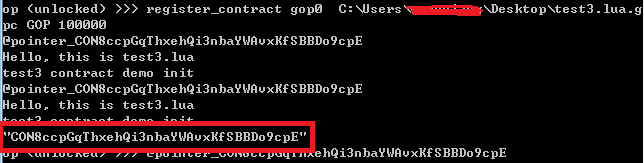
* **命令行命令**：register\_contract
* **参数**：(1) 注册者账户名

(2) 合约文件编译生成的字节码文件

(3) 资产标识

(4) 合约初始化限制

* **返回值**：合约的地址
* **示例**：

****

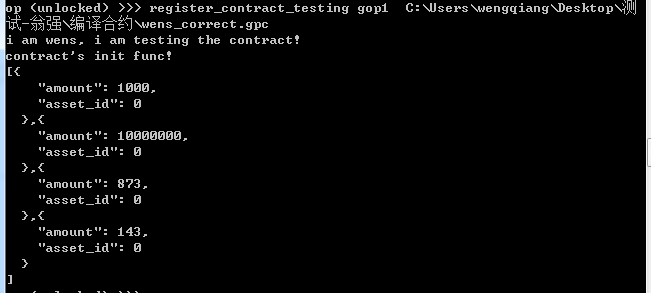
### 注册合约(仅本地)

此命令跟上面的注册合约(链上)的命令是相对的， 上面的命令是会将合约注册到区块链上， 而此命令仅作本地试验或想看看效果时用(后面的带链上和仅本地同名命令的与此类同).

* **命令行命令:** register\_contract\_testing
* **参数:** (1) 注册者账户名

(2) 合约文件编译生成的字节码文件

* **返回值:** std::vector<goopal::blockchain::Asset>
* **示例:**

****

### 调用合约(链上)

* **命令行命令:** call\_contract
* **参数:** (1) 合约地址（或者合约名称）

(2) 合约调用者账户名

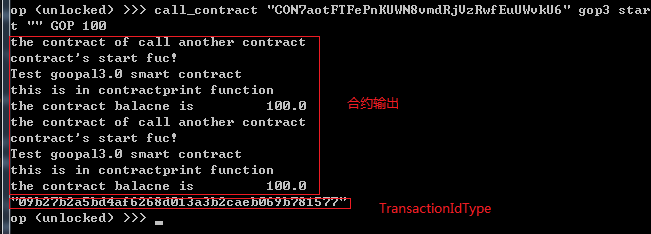
(3) 合约中要调用的方法名

(4) 要调用的合约方法参数

(5) 资产标识

(6) 调用费用的限制

* **返回值:** TransactionIdType
* **示例:**

****

### 调用合约(仅本地)

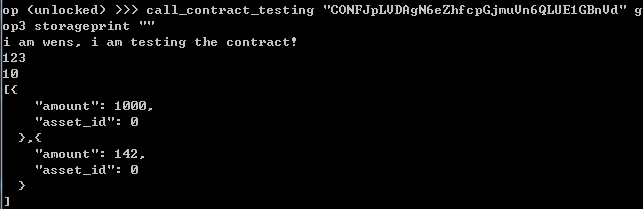
* **命令行命令:** call\_contract\_testing
* **参数:** (1) 合约地址（或者合约名称）

(2) 合约调用者账户名

(3) 合约中要调用的方法名

(4) 要调用的合约方法参数

* **返回值:** std::vector<goopal::blockchain::Asset>
* **示例:**

****

### 升级合约

* **命令行命令:** upgrade\_contract
* **参数:** (1) 合约地址

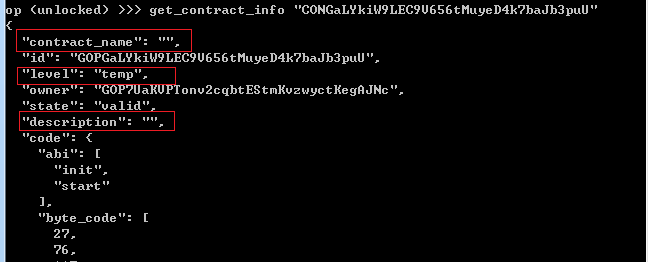
(2) 升级者账户名

(3) 合约名字

(4) 合约描述

* **返回值:** WalletTransactionEntry
* **示例:**

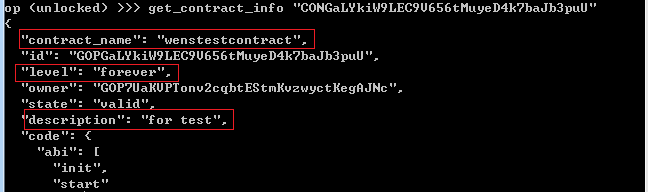
首先查看未升级的合约(temp)的信息：



升级合约：



最后查看升级后的合约信息：



### 销毁合约

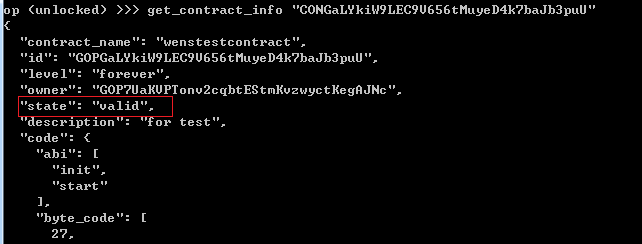
* **命令行命令:** destroy\_contract
* **参数:** (1) 合约地址

(2) 销毁者账户名

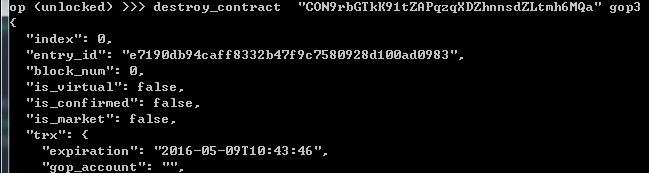
* **返回值:** WalletTransactionEntry
* **示例:**

只能销毁临时合约(状态为temp)的合约，销毁永久合约(状态为forever)会给出提示信息.

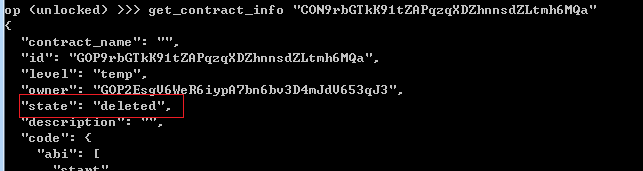
销毁之前:



销毁合约：



销毁之后：



### 转账到合约(链上)

* **命令行命令:** wallet\_transfer\_to\_contract
* **参数:** (1) 转账到合约balance的金额

(2) 资产标识

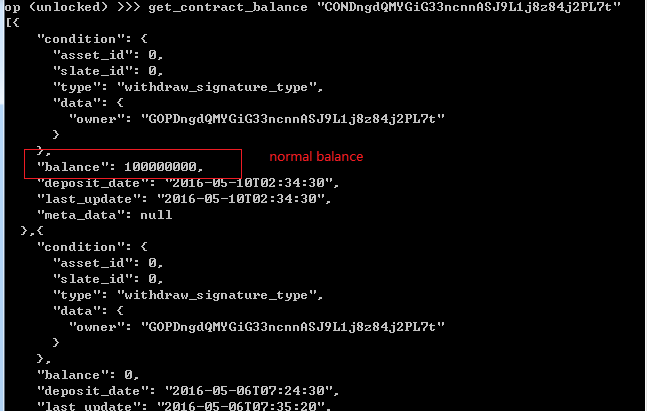
(3) 转账的账户名

(4) 合约地址(或者合约名称)

(5) 执行触发函数的金额

* **返回值:** WalletTransactionEntry
* **示例:**

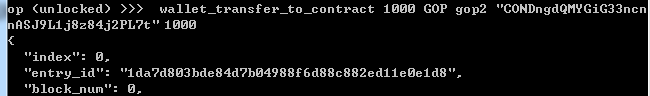
首先查看合约balance下的初始金额



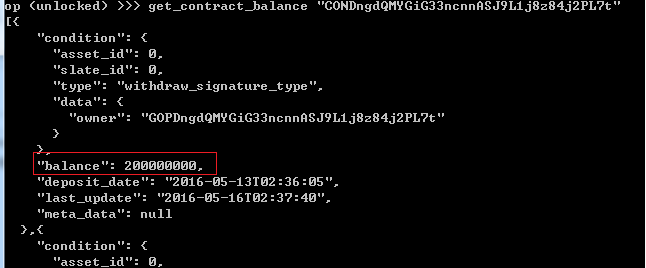
然后查看转账的账户的原始金额



接着转账到合约(这里以地址为例)



最后看合约balance的金额(显示的是1000 \* 100000)



账户余额 ( 被转账 1000 加上 0.01的手续费)



### 转账到合约(仅本地)

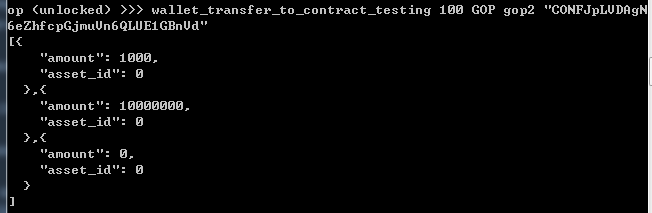
* **命令行命令:** wallet\_transfer\_to\_contract\_testing
* **参数:** (1) 转账到合约balance的金额

(2) 资产标识

(3) 转账的账户名

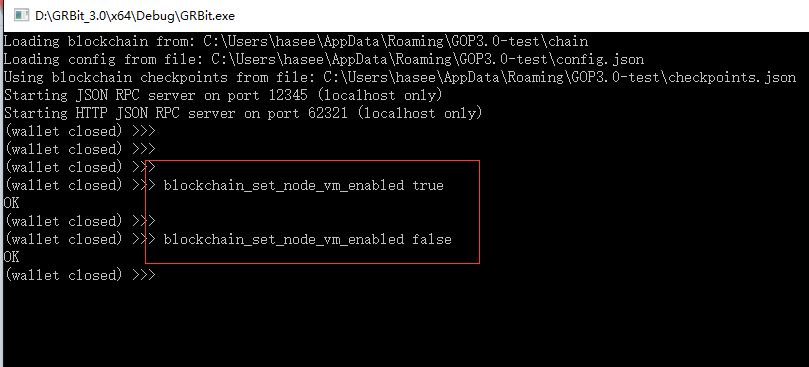
(4) 合约地址(或者合约名称)

* **返回值:** std::vector<goopal::blockchain::Asset>
* **示例:**



### 打开/关闭本地解释器验证开关（仅对普通节点有效）

* **命令行命令:** blockchain\_set\_node\_vm\_enabled
* **参数:** (1) 打开/关闭
* **返回值: 无**
* **示例:**

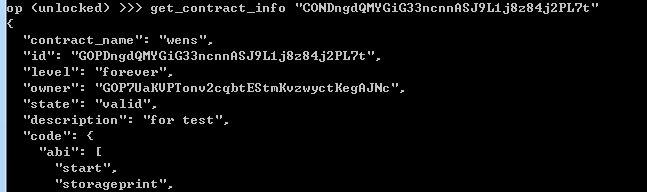


## 接口功能API：

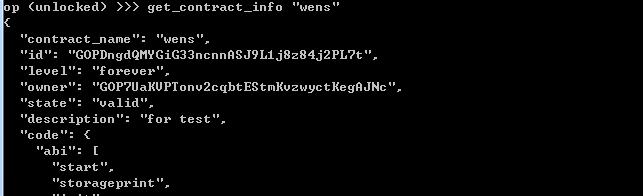
### 根据合约地址/名字获取合约信息

* **命令行命令:** get\_contract\_info
* **参数:** (1) 合约地址（或者合约名字）
* **返回值:** ContractEntryPrintable
* **示例:**

根据地址:

****

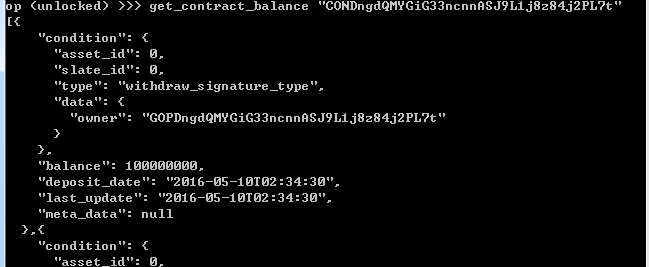
根据合约名称:



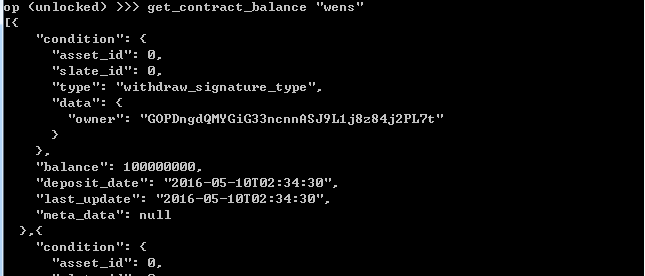
### 根据合约地址/名字获取合约balance

* **命令行命令:** get\_contract\_balance
* **参数:** (1) 合约地址（或者合约名字）
* **返回值:** vector<goopal::blockchain::BalanceEntry>
* **示例:**

根据地址:

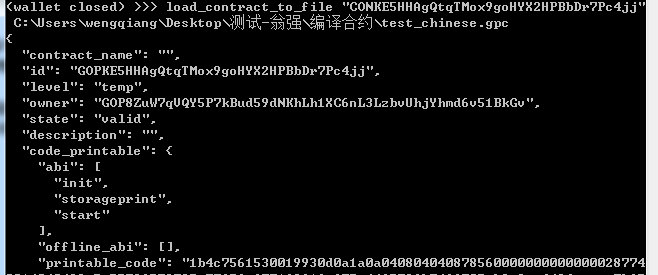


根据名称:



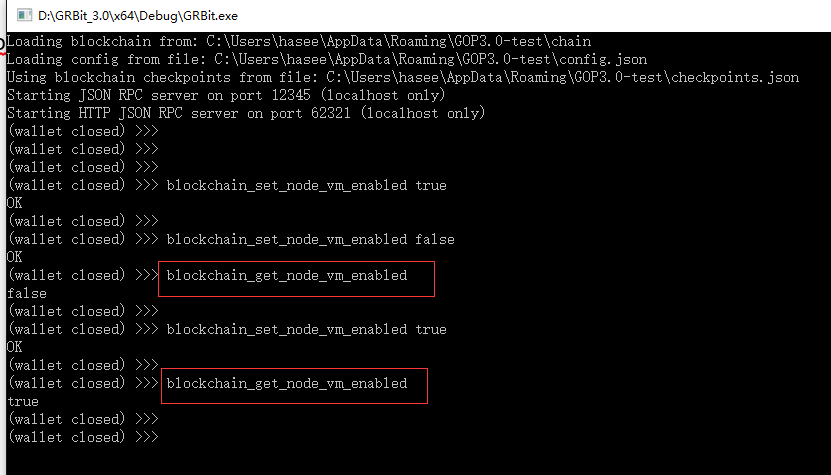
### 导出合约到文件

* **命令行命令:** load\_contract\_to\_file
* **参数:** (1) 合约地址（或者合约名字）
* (2) 带路径的文件名(用于保存导出的合约)
* **返回值:** ContractEntryPrintable
* **示例:**



### 获取是否打开本地解释器验证（仅对普通节点有效）

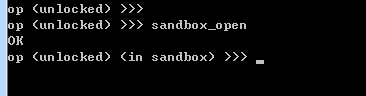
* **命令行命令:** blockchain\_get\_node\_vm\_enabled
* **参数:** 无
* **返回值:**  Bool类型
* **示例:**



## Sandbox功能API

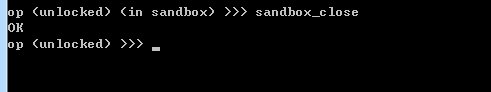
### 打开沙盒

* **命令行命令:** sandbox\_open
* **参数:** 无
* **返回值:** void
* **示例:**



### 关闭沙盒

* **命令行命令:** sandbox\_close
* **参数:** 无
* **返回值:** void
* **示例:**



### 沙盒中注册合约

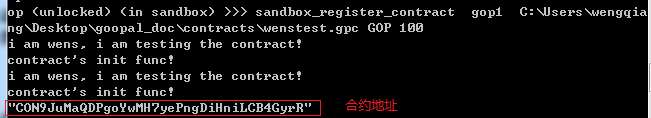
* **命令行命令**：sandbox\_register\_contract
* **参数**：(1) 注册者账户名

(2) 合约文件编译生成的字节码文件

(3) 资产标识

(4) 合约初始化限制

* **返回值**：合约的地址
* **示例**：

****

### 沙盒中调用合约

* **命令行命令:** sandbox\_call\_contract
* **参数:** (1) 合约地址（或者合约名称）

(2) 合约调用者账户名

(3) 合约中要调用的方法名

(4) 要调用的合约方法参数

(5) 资产标识

(6) 调用费用的限制

* **返回值:** TransactionIdType
* **示例:**

****

### 沙盒中升级合约

* **命令行命令:** sandbox\_upgrade\_contract
* **参数:** (1) 合约地址

(2) 升级者账户名

(3) 合约名字

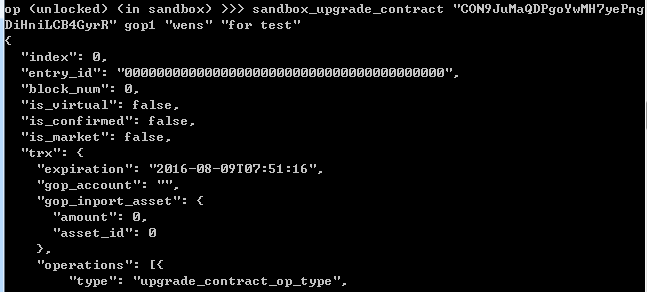
(4) 合约描述

* **返回值:** WalletTransactionEntry
* **示例:**

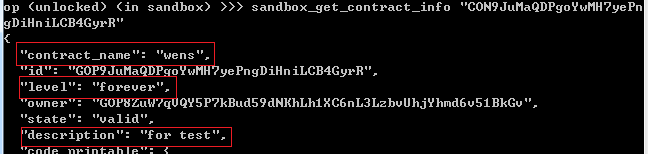
首先查看未升级的合约(temp)的信息：



升级合约：



最后查看升级后的合约信息：



### 沙盒中销毁合约

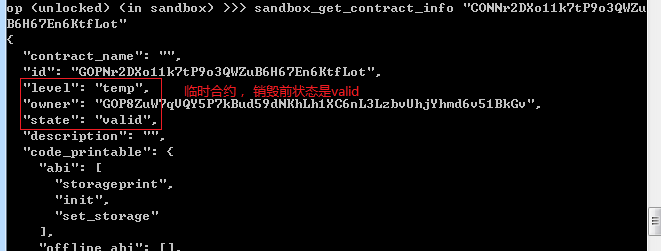
* **命令行命令:** sandbox\_destroy\_contract
* **参数:** (1) 合约地址

(2) 销毁者账户名

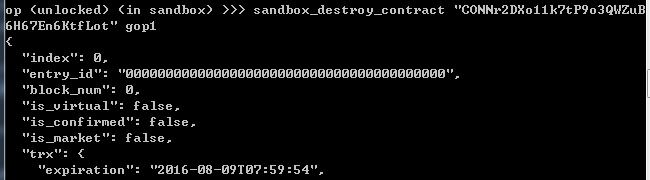
* **返回值:** WalletTransactionEntry
* **示例:**

只能销毁临时合约(状态为temp)的合约，销毁永久合约(状态为forever)会给出提示信息.

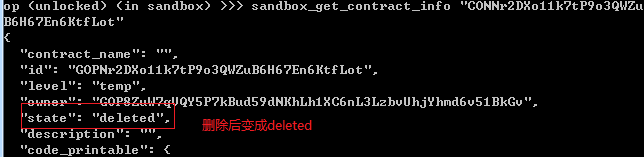
销毁之前:



销毁合约：



销毁之后：



### 沙盒中转账到合约

* **命令行命令:** sandbox\_wallet\_transfer\_to\_contract
* **参数:** (1) 转账到合约balance的金额

(2) 资产标识

(3) 转账的账户名

(4) 合约地址(或者合约名称)

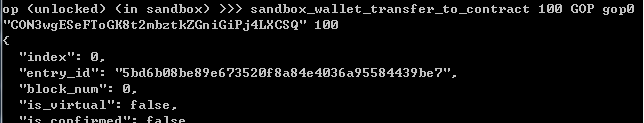
(5) 执行触发函数的金额

* **返回值:** WalletTransactionEntry
* **示例:**

首先查看合约balance下的初始金额



接着转账到合约(这里以地址为例)



最后看合约balance的金额(显示的是100 \* 100000)



### 沙盒中获取合约信息

* **命令行命令:** sandbox\_get\_contract\_info
* **参数:** (1) 合约地址（或者合约名字）
* **返回值:** ContractEntryPrintable
* **示例:**

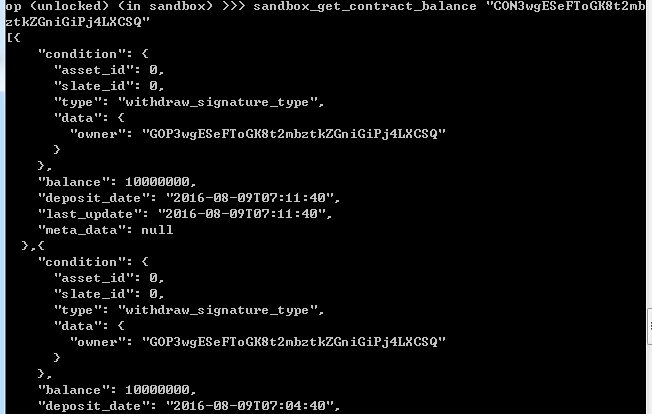
同样可以根据地址或合约名，以下以地址为例:

****

### 沙盒中获取合约balance

* **命令行命令:** sandbox\_get\_contract\_balance
* **参数:** (1) 合约地址（或者合约名字）
* **返回值:** vector<goopal::blockchain::BalanceEntry>
* **示例:**

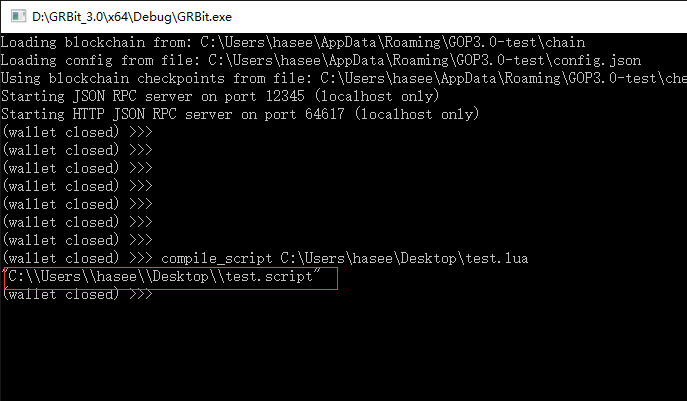
同样可以根据地址或合约名，以下以地址为例:



## Event功能API：

### 编译本地脚本

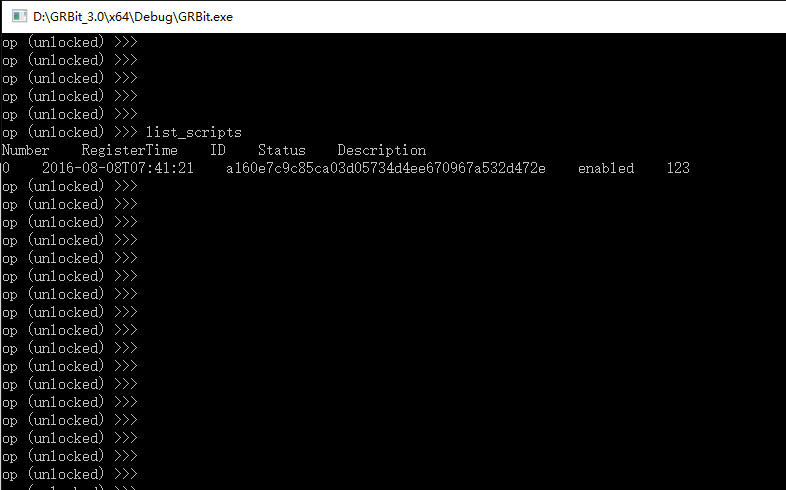
* **命令行命令:** compile\_script
* **参数:** (1) 待编译的脚本文件（文件为lua源码文件，文件后缀为.lua）
* **返回值:** 生成的脚本字节码文件，文件后缀为.script
* **示例:**



### 列出打开钱包中的所有本地脚本

* **命令行命令:** list\_scripts
* **参数:** 无
* **返回值:** vector<ScriptEntryPrintable>

**示例:**



### 查询打开钱包中的指定的本地脚本

* **命令行命令:** get\_script\_info
* **参数:** (1) 脚本ID
* **返回值:** Optional<ScriptEntryPrintable>

**示例:**



### 在钱包中添加新的本地脚本

* **命令行命令:** add\_script
* **参数:** (1) 本地脚本的绝对路径
* **返回值:** ScriptIdType

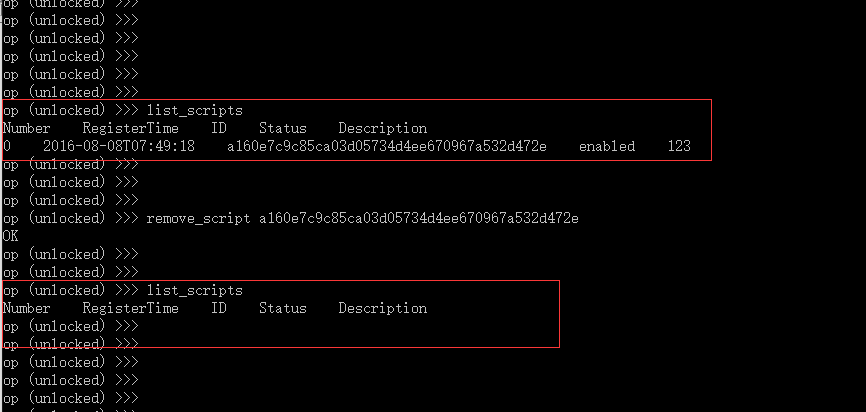
**示例:**



### 从钱包中删除本地脚本

* **命令行命令:** remove\_script
* **参数:** (1) 脚本ID
* **返回值: 无**

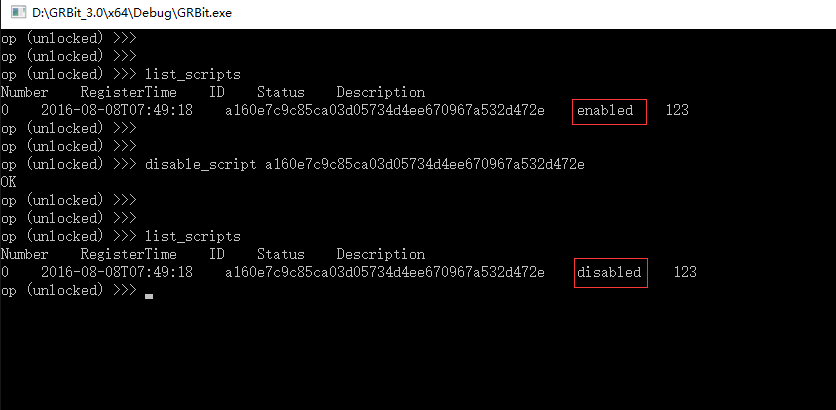
**示例:**



### 禁用钱包中的一个本地脚本

* **命令行命令:** disable\_script
* **参数:** (1) 脚本ID
* **返回值: 无**

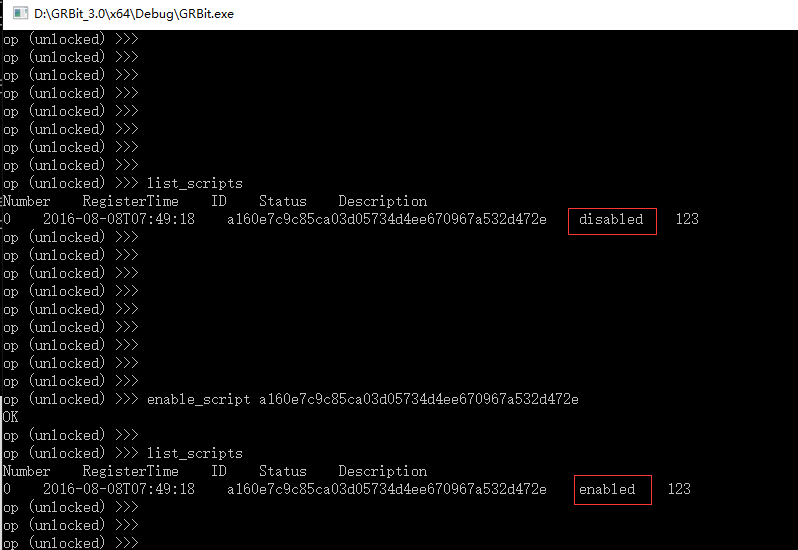
**示例:**



### 启用钱包中的一个本地脚本

* **命令行命令:** enable\_script
* **参数:** (1) 脚本ID
* **返回值: 无**

**示例:**



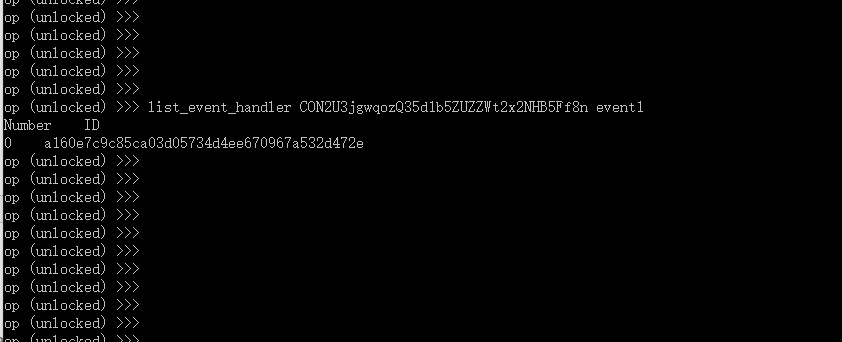
### 查询与合约中的Event相关的本地脚本

* **命令行命令:** list\_event\_handler
* **参数:** (1) 相关的合约地址

(2) 相关的Event类型

* **返回值:** vector<ScriptIdType>

**示例:**



### 向合约中的Event添加一个相关的本地脚本

* **命令行命令:** add\_event\_handler
* **参数:** (1) 相关的合约地址

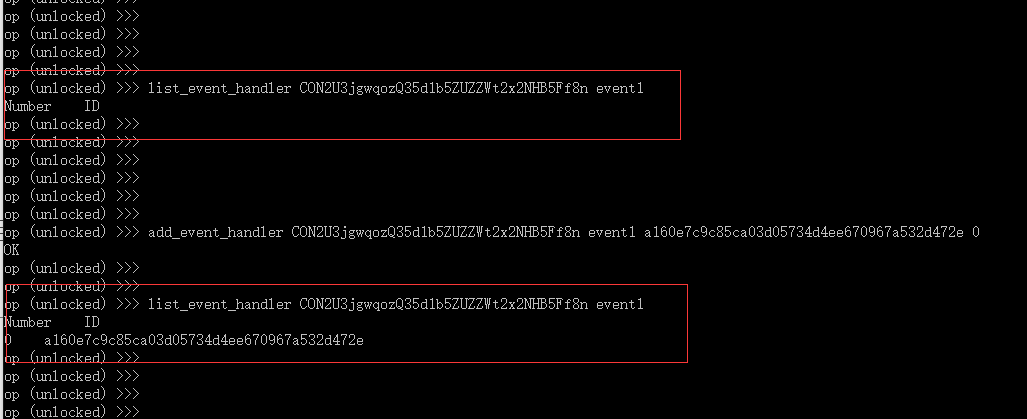
(2) 相关的Event类型

(3) 脚本ID

(4) 插入序号

* **返回值: 无**

**示例:**



### 从合约中的Event删除一个相关的本地脚本

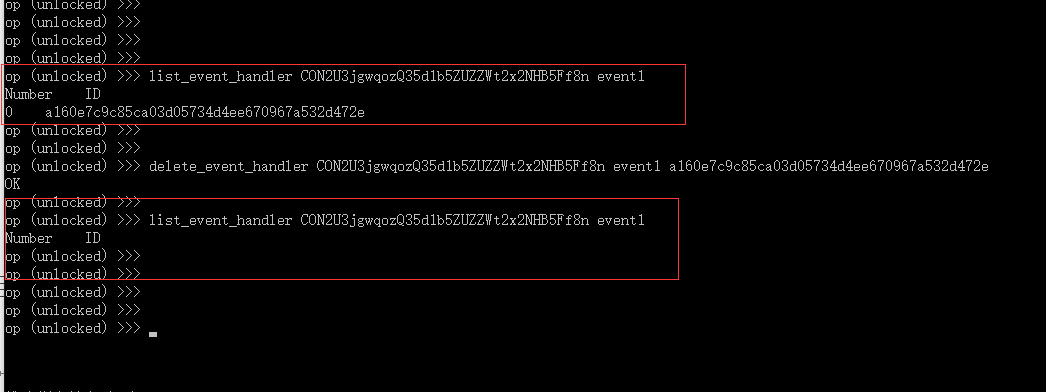
* **命令行命令:** delete\_event\_handler
* **参数:** (1) 相关的合约地址

(2) 相关的Event类型

(3) 脚本ID

* **返回值: 无**

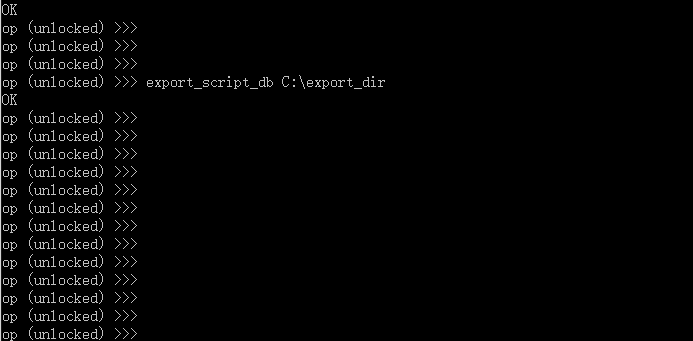
**示例:**

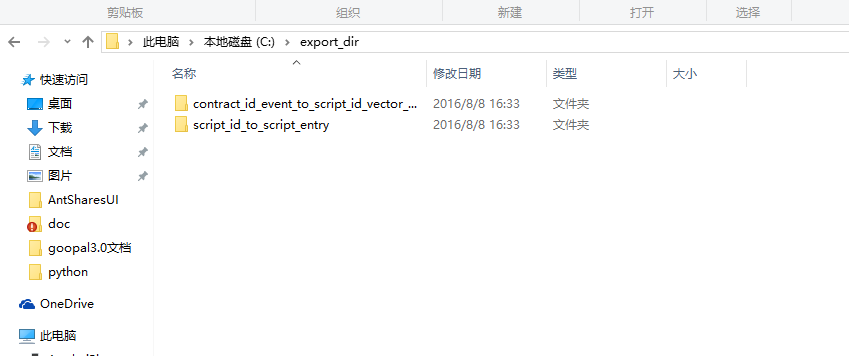


### 导出本地脚本库以及与Event的关系库

* **命令行命令:** export\_script\_db
* **参数:** (1) 需要导出的库的目标目录
* **返回值: 无**

**示例:**





### 导入本地脚本库以及与Event的关系库

* **命令行命令:** import\_script\_db
* **参数:** (1) 需要导入的库的源目录
* **返回值: 无**

**示例:**

