**EOS Command Line Client** (**eosc**)

**Tool for sending transactions and querying state from eosd. More...**

**EOS**命令行客户端（**eosc**）

用于从**eosd**发送事务和查询状态的工具。[更多](https://eosio.github.io/eos/group__eosc.html" \l "details)**[...](https://eosio.github.io/eos/group__eosc.html" \l "details)**

**Introduction to EOSC**

**eosc is a command line tool that interfaces with the REST api exposed by eosd. In order to use eosc you will need to have a local copy of eosd running and configured to load the 'eos::chain\_api\_plugin'.**

# Plugin(s) to enable, may be specified multiple times

 plugin = eos::producer\_plugin

 plugin = eos::chain\_api\_plugin

**EOSC**简介

eosc是用来与eosd 提供的REST api接口进行交互的命令行工具。为了使用eosc，需要在本地运行的 eosd，并且已配置加载 'eos :: chain\_api\_plugin' 插件。

＃启用的插件，可以指定多个

 plugin = eos::producer\_plugin

 plugin = eos::chain\_api\_plugin

**After starting eosd you should be able to query the current blockchain state like so:**

./eosc info

{

 "head\_block\_num": 25048,

 "last\_irreversible\_block\_num": 25027,

 "head\_block\_id": "000061d8ae49d6af614e02779e19261959f22d6d9f37906ed5db2dabd88be142",

 "head\_block\_time": "2017-07-25T17:44:48",

 "head\_block\_producer": "initi",

 "recent\_slots": "1110000000000000000000000000000000000000000000000000000000000011",

 "participation\_rate": "0.07812500000000000"

}

启动eosd后，就可以查询当前的区块状态，如下所示：

./eosc info

{

 "head\_block\_num": 25048,

 "last\_irreversible\_block\_num": 25027,

 "head\_block\_id": "000061d8ae49d6af614e02779e19261959f22d6d9f37906ed5db2dabd88be142",

 "head\_block\_time": "2017-07-25T17:44:48",

 "head\_block\_producer": "initi",

 "recent\_slots": "1110000000000000000000000000000000000000000000000000000000000011",

 "participation\_rate": "0.07812500000000000"

}

**Creating an Account**

**In order to create an account you will need two new keys: owener and active. You can ask eosc to create some keys for you:**

./eosc create key

public: EOS4toFS3YXEQCkuuw1aqDLrtHim86Gz9u3hBdcBw5KNPZcursVHq

private: 5JKbLfCXgcafDQVwHMm3shHt6iRWgrr9adcmt6vX3FNjAEtJGaT

创建账户

为了创建一个帐户，您需要两个新的密钥：owener和active。您可以使用eosc创建密钥：

./eosc create key

public: EOS4toFS3YXEQCkuuw1aqDLrtHim86Gz9u3hBdcBw5KNPZcursVHq

private: 5JKbLfCXgcafDQVwHMm3shHt6iRWgrr9adcmt6vX3FNjAEtJGaT

**Note**

**eosc does not save the generated private key.**

**Next we will create the account tester**, **but because all accounts need to be created by an existing account we will ask the inita account to create tester. inita was specified in the genesis file.**

注意

eosc不保存生成的私钥。

接下来，我们将创建该帐户tester，但是由于所有帐户都需要由现有帐户创建，我们将要求inita帐户创建tester。inita用户由创世块配置文件设定。

./eosc create account inita tester EOS4toFS3YXEQCkuuw1aqDLrtHim86Gz9u3hBdcBw5KNPZcursVHq EOS6KdkmwhPyc2wxN9SAFwo2PU2h74nWs7urN1uRduAwkcns2uXsa

{

  "transaction\_id": "6acd2ece68c4b86c1fa209c3989235063384020781f2c67bbb80bc8d540ca120",

  "processed": {

    "refBlockNum": "25217",

    "refBlockPrefix": "2095475630",

    "expiration": "2017-07-25T17:54:55",

    "scope": [

      "eos",

      "inita"

    ],

    "signatures": [],

    "messages": [{

        "code": "eos",

        "type": "newaccount",

        "authorization": [{

            "account": "inita",

            "permission": "active"

          }

        ],

        "data": "c9251a0000000000b44c5a2400000000010000000102bcca6347d828d4e1868b7dfa91692a16d5b20d0ee3d16a7ca2ddcc7f6dd03344010000010000000102bcca6347d828d4e1868b7dfa91692a16d5b20d0ee3d16a7ca2ddcc7f6dd03344010000010000000001c9251a000000000061d0640b000000000100010000000000000008454f5300000000"

      }

    ],

    "output": [{

        "notify": [],

        "sync\_transactions": [],

        "async\_transactions": []

      }

    ]

  }

}

**After creating the account we can view the current account status like so:**

创建帐户后，我们可以查看当前帐户状态：

./eosc account tester

{

  "name": "tester",

  "eos\_balance": 0,

  "staked\_balance": 1,

  "unstaking\_balance": 0,

  "last\_unstaking\_time": "1969-12-31T23:59:59"

}

**You will note that there is no balance because almost all genesis EOS tokens are currently allocated to the eos account.**

你可能会注意到，帐户余额为0，因为几乎所有的EOS代币目前都被分配到eos帐户中。

./eosc account [eos](https://eosio.github.io/eos/namespaceeos.html)

{

  "name": "eos",

  "eos\_balance": "8999999999998100",

  "staked\_balance": 0,

  "unstaking\_balance": 0,

  "last\_unstaking\_time": "1969-12-31T23:59:59",

  "abi": {

    "types": [{

        "newTypeName": "AccountName",

        "type": "Name"

      }

    ],

    "structs": [{

        "name": "transfer",

        "base": "",

        "fields": {

          "from": "AccountName",

          "to": "AccountName",

          "amount": "UInt64"

        }

      }

    ],

    "actions": [{

        "action": "transfer",

        "type": "transfer"

      }

    ],

    "tables": []

  }

}

**Note**

**The eos account happens to have an ABI** (**Application Binary Interface**) **defined which provides meta-data to tools that want to interface with the eos contract.**

**We can fund our tester account via eosc with the following command:**

注意

该eos帐户刚好配有一个ABI（应用程序二进制接口），它提供元数据给eos智能合约交互工具。

使用 eosc 命令可以给 tester 帐户充值：

./eosc transfer [eos](https://eosio.github.io/eos/namespaceeos.html) tester 1000

{

  "transaction\_id": "52b488d27ce1f72a2b29f22e5e1638fa5db5d7805565884e795733a15c6c2195",

  "processed": {

    "refBlockNum": "25298",

    "refBlockPrefix": "1151709320",

    "expiration": "2017-07-25T17:58:58",

    "scope": [

      "eos",

      "tester"

    ],

    "signatures": [],

    "messages": [{

        "code": "eos",

        "type": "transfer",

        "authorization": [{

            "account": "eos",

            "permission": "active"

          }

        ],

        "data": {

          "from": "eos",

          "to": "tester",

          "amount": 1000

        },

        "hex\_data": "e54d000000000000b44c5a2400000000e803000000000000"

      }

    ],

    "output": [{

        "notify": [{

            "name": "tester",

            "output": {

              "notify": [],

              "sync\_transactions": [],

              "async\_transactions": []

            }

          }

        ],

        "sync\_transactions": [],

        "async\_transactions": []

      }

    ]

  }

}

**Now we can verify that the funds were received.**

现在我们可以验证收到的资金。

./eosc account tester

{

  "name": "tester",

  "eos\_balance": 1000,

  "staked\_balance": 1,

  "unstaking\_balance": 0,

  "last\_unstaking\_time": "1969-12-31T23:59:59"

}

**Creating a Contract**

**In this section we will use eosc to create and publish a currency contract. You can find the example currency contract in the eos/contracts/currency directory.**

**The first step is to create an account for currency. We will have the tester account create the currency account.**

创建合约

在本节中，我们将用于eosc创建和发布智能合约。示例在eos/contracts/currency目录中。

第一步是创建合约账号。我们将使用tester帐户创建currency帐户。

./eosc create account tester [currency](https://eosio.github.io/eos/namespacecurrency.html) EOS4toFS3YXEQCkuuw1aqDLrtHim86Gz9u3hBdcBw5KNPZcursVHq EOS6KdkmwhPyc2wxN9SAFwo2PU2h74nWs7urN1uRduAwkcns2uXsa

**The next step is to publish the contract** (**.wast**) **and its abi** (**.abi**)

下一步是发布合约（.wast）及其abi（.abi）

./eosc setcode [currency](https://eosio.github.io/eos/namespacecurrency.html) ../../contracts/[currency](https://eosio.github.io/eos/namespacecurrency.html)/[currency](https://eosio.github.io/eos/namespacecurrency.html).wast ../../contracts/[currency](https://eosio.github.io/eos/namespacecurrency.html)/[currency](https://eosio.github.io/eos/namespacecurrency.html).abi

{

  "transaction\_id": "738669518a8fc6935394992beec1dc4dc1b60f7d9f232b6ccd6a282619eedca9",

  "processed": {

    "refBlockNum": "25386",

    "refBlockPrefix": "154808726",

    "expiration": "2017-07-25T18:03:22",

    "scope": [

      "eos",

      "currency"

    ],

    "signatures": [],

    "messages": [{

        "code": "eos",

        "type": "setcode",

        "authorization": [{

            "account": "currency",

            "permission": "active"

          }

        ],

        "data": ""

      }

    ],

    "output": [{

        "notify": [],

        "sync\_transactions": [],

        "async\_transactions": []

      }

    ]

  }

}

**After the contract is published it initially allocates all of the currency to the currency account. So lets transfer some of it to our tester.**

合约发布后，它首先将所有代币分配到currency帐户。所以我们先转移一些代币到tester帐户上。

./eosc exec [currency](https://eosio.github.io/eos/namespacecurrency.html) transfer '{"from":"currency","to":"tester","amount":50}' '["currency","tester"]' '[{"account":"currency","permission":"active"}]'

{

  "transaction\_id": "f601f2fdb26e366a19913229e4d2928778b50166811c63c7962401b11d23ef3d",

  "processed": {

    "refBlockNum": "25427",

    "refBlockPrefix": "2231248056",

    "expiration": "2017-07-25T18:05:25",

    "scope": [

      "tester",

      "currency"

    ],

    "signatures": [],

    "messages": [{

        "code": "currency",

        "type": "transfer",

        "authorization": [{

            "account": "currency",

            "permission": "active"

          }

        ],

        "data": {

          "from": "currency",

          "to": "tester",

          "amount": 50

        },

        "hex\_data": "a34a59dcc8000000b44c5a24000000003200000000000000"

      }

    ],

    "output": [{

        "notify": [{

            "name": "tester",

            "output": {

              "notify": [],

              "sync\_transactions": [],

              "async\_transactions": []

            }

          }

        ],

        "sync\_transactions": [],

        "async\_transactions": []

      }

    ]

  }

}

**The exec command takes the following arguments:**

* code - the account whose contract code should be run
* action - the type of the message to pass to code
* data - a JSON blob (or hex string) of the message data as defined by the ABI for the action type.
* scope - a JSON array of account names which contain data that may be read or modified by code (in this case the sender and receiver)
* authorization - the account and permission level which authorized the action

该exec命令采用以下参数：

* code - 需要运行合约的帐户 需要运行的合约帐户？
* action - 传递给帐户的消息类型
* data - 由ABI为消息类型定义的消息数据，使用JSON blob（或十六进制字符串）格式。
* scope - 包含可以被code参数（在本例中为发送者和接收者）读取或修改的的帐户名称的JSON格式数组，
* authorization - 授权操作的帐户和权限级别

**Note**

**at this time the blockchain is not validating signatures so anyone can do anything provided they simply declare the proper authority. In the future this will direct the wallet on which keys to use to sign it. Also future revisions of this API may automatically detect scope and authorization via a trial run of the contract.**

注意

**目前，区块链没有验证签名，所以所有帐户都可以做任何事情，只要他们声明有相应的权限。以后，将直接使用钱包中的私钥签名。此API的未来版本也可能会通过合约的试运行来自动检测授权和范围。**