PROJECT 9
GETTING STARTED WITH MULTICHAIN
SPRING TRIMESTER 2022

By
NIKHIL PATEL

OVERVIEW :- In this lab we will figure out how to introduce multichain server and will likewise alter the blockchain. Then we will create addresses to get a resource and give admittance to the administrator so he have some control over a whole chain to who can get the resource and who can send and can interface with the resource. We took in a cooperative mining strategy(robin round) to mine a couple of blocks.

Multichain establishment is speedy and simple to arrange likewise when order execute as a reaction it determines what definite order ought to run on opposite side server to make association or on the other hand on the off chance that having a mistake, give precise answer for that in reaction.
Whole multichain is constrained by proprietor who made multichain or he can give authorization to other who can send resource and get additionally for interfacing multichain. We can get exchange on each host which is associated with multichain.We can send resource with metadata; metadata is utilized for depicting for what reason resources had been send ex abc installment. We can send nuclear exchange implies the two sides probably succeeded or bombed exchanges.

PROCEDURE :- The steps for this project are as follows :

Step 1: Initially we have to update our machine by executing following command :-

```
sudo apt-get update
cd /tmp
wget http://www.multichain.com/download/multichain-1.0.1.tar.gz
tar -xvzf multichain-1.0.1.tar.gz
cd multichain-1.0.1
sudo mv multichaind multichain-cli multichain-util /usr/local/bin
```

Now, you have to choose one of the virtual machines as First Server(Multichain 1) & other vm as second server (Multichain 2).

multichain-1.0.1 : bash — Konsole

File Edit View Bookmarks Settings Help

```
elzorro@ubuntu:/tmp/multichain-1.0.1$ clear
elzorro@ubuntu:/tmp/multichain-1.0.1$ sudo apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu xenial-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu xenial-security InRelease
Reading package lists... Done
elzorro@ubuntu:/tmp/multichain-1.0.1$ cd /tmp
elzorro@ubuntu:/tmp$ wget http://www.multichain.com/download/multichain-1.0.1.tar.gz
--2022-07-09 15:43:46--  http://www.multichain.com/download/multichain-1.0.1.tar.gz
Resolving www.multichain.com (www.multichain.com)... 162.243.214.85
Connecting to www.multichain.com (www.multichain.com)|162.243.214.85|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.multichain.com/download/multichain-1.0.1.tar.gz [following]
--2022-07-09 15:44:01--  https://www.multichain.com/download/multichain-1.0.1.tar.gz
Connecting to www.multichain.com (www.multichain.com)|162.243.214.85|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10055511 (9.6M) [application/x-gzip]
Saving to: 'multichain-1.0.1.tar.gz.1'

multichain-1.0.1.tar. 100%[=========================>]   9.59M   366KB/s    in 31s

2022-07-09 15:44:32 (321 KB/s) - 'multichain-1.0.1.tar.gz.1' saved [10055511/10055511]

elzorro@ubuntu:/tmp$ tar -xvzf multichain-1.0.1.tar.gz
multichain-1.0.1/
multichain-1.0.1/multichain-util
multichain-1.0.1/multichain-cli
multichain-1.0.1/README.txt
multichain-1.0.1/multichaind
multichain-1.0.1/multichaind-cold
elzorro@ubuntu:/tmp$ cd multichain-1.0.1
elzorro@ubuntu:/tmp/multichain-1.0.1$ sudo mv multichaind multichain-cli multichain-util
 /usr/local/bin
elzorro@ubuntu:/tmp/multichain-1.0.1$ multichain-util create chain1
```

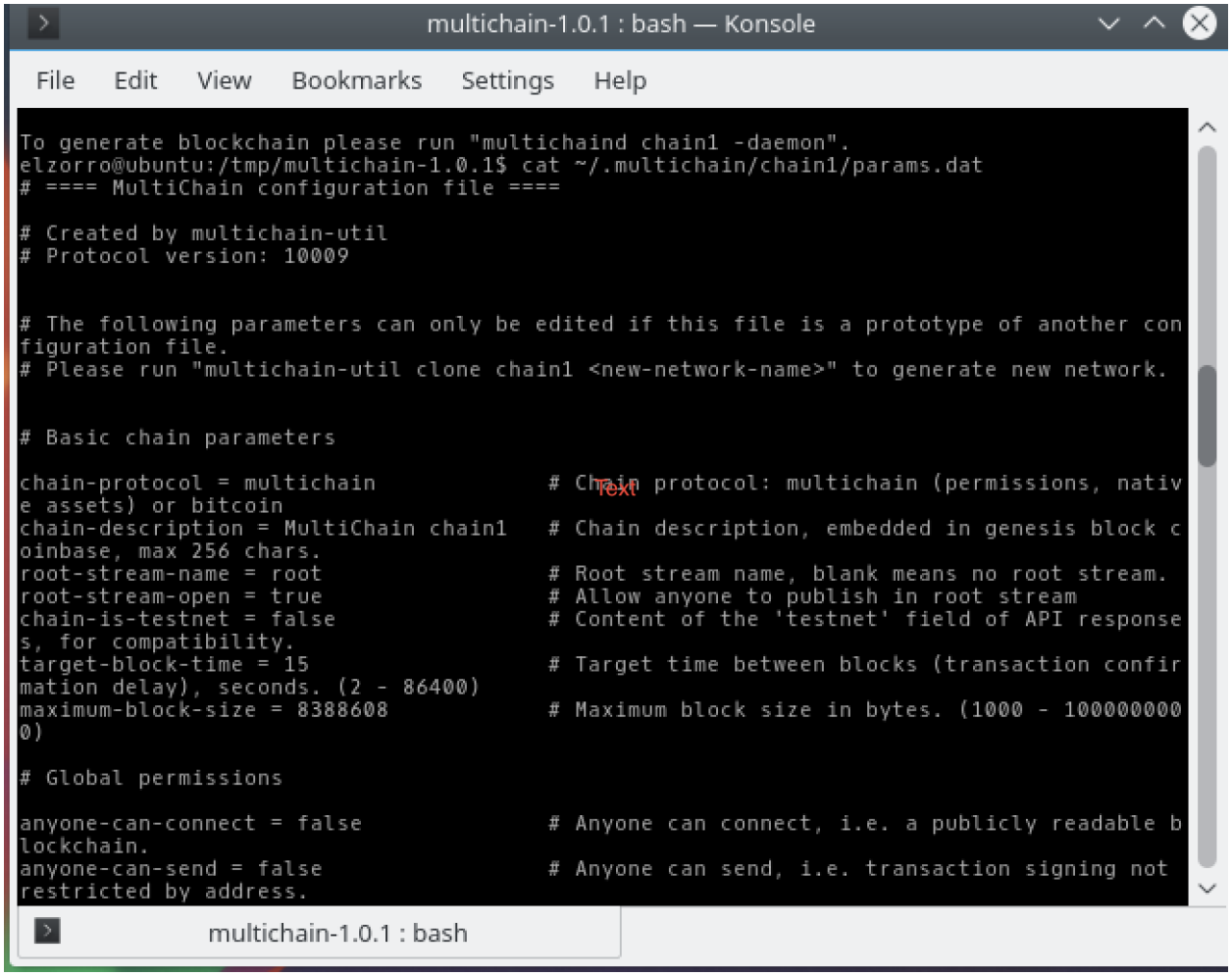
multichain-1.0.1 : bash

Step 2 : We will now create a new blockchain server named chain 1.

multichain-util create chain1 (Server 1)

To view the blockchain default setting enter :

cat ~/.multichain/chain1/params.dat (Server 1)



Step 3: Now we will initialize the blockchain and execute the command.Wait for a few seconds to load the server and after a few seconds you will get the message genesis block found. You will also get the node address so that others can use that address to connect with you.

multichaind chain1 –daemon (Server 1)

Wait for few second to load the server and after few second the genesis block will

Step 4: Make note of node address as shown below :

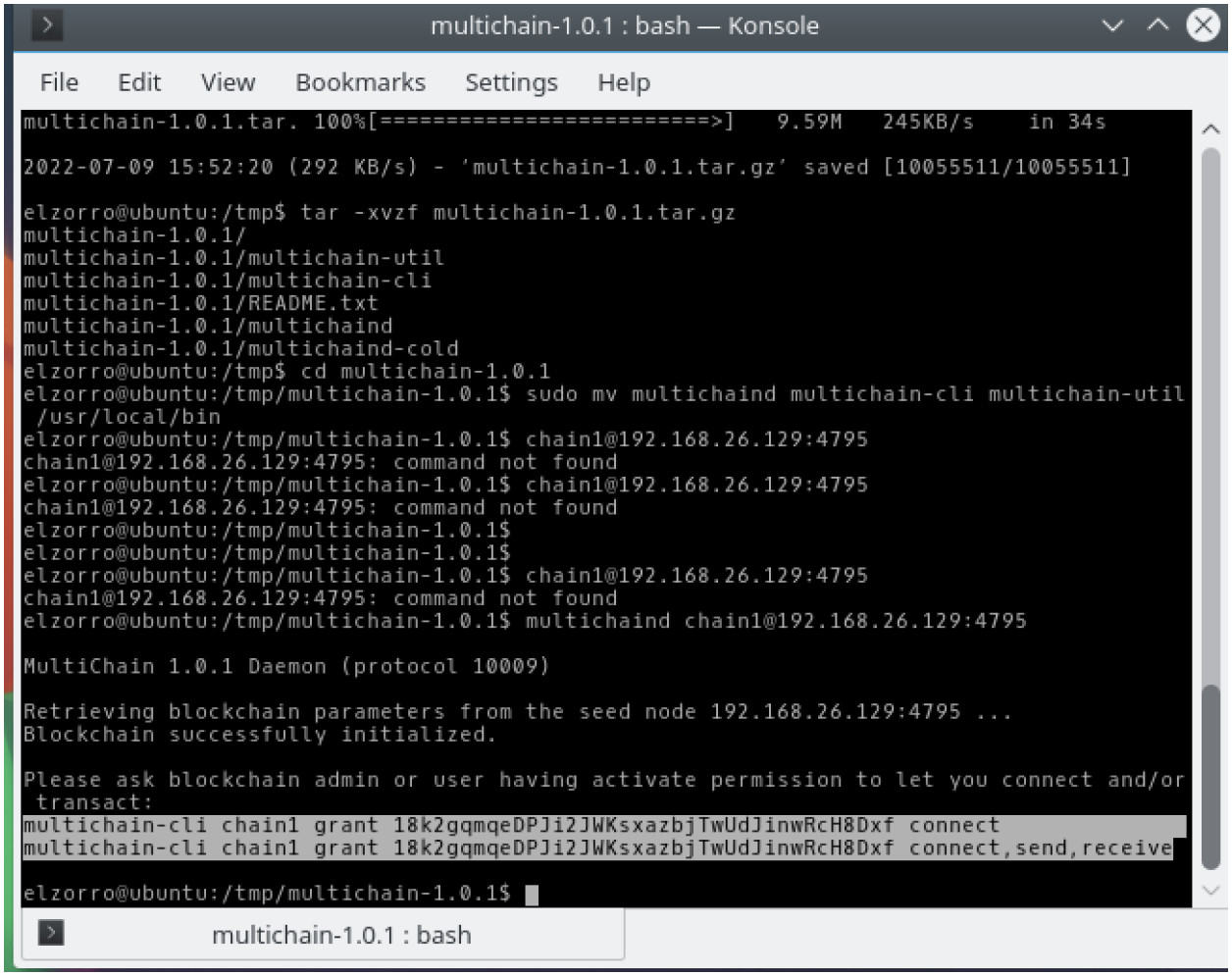
chain1@192.168.227.132:2771 (Your ipaddress will be different) **[SERVER 2]**

Step 5: Now we will try to connect with the blockchain from a different server i.e. its second server multichain 2.

multichaind chain1@192.168.227.132:2771

Note down the address of wallet, you can check in the end of the multichain-cli command as shown below

1bXopFL2VzG4vrA3v89rqGK5XWyYWi7r8v46XP

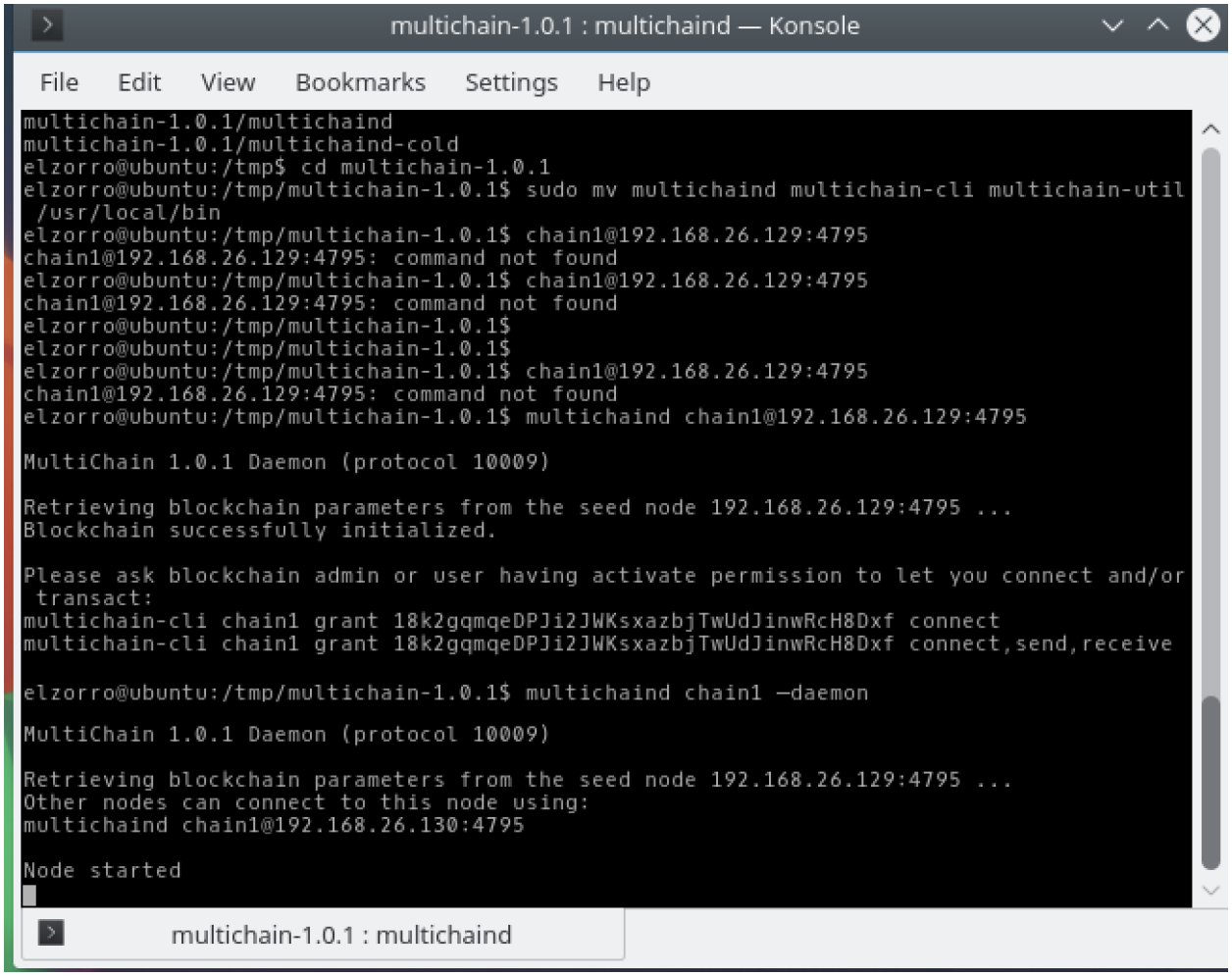


Step 6: Now we will go back to server 1 and add replace the wallet address with your own address received from second server:

multichain-cli chain1 grant " *wallet address* "connect (Server 1)

Step 7: Now try to reconnect the second server by entering the following command in the console.

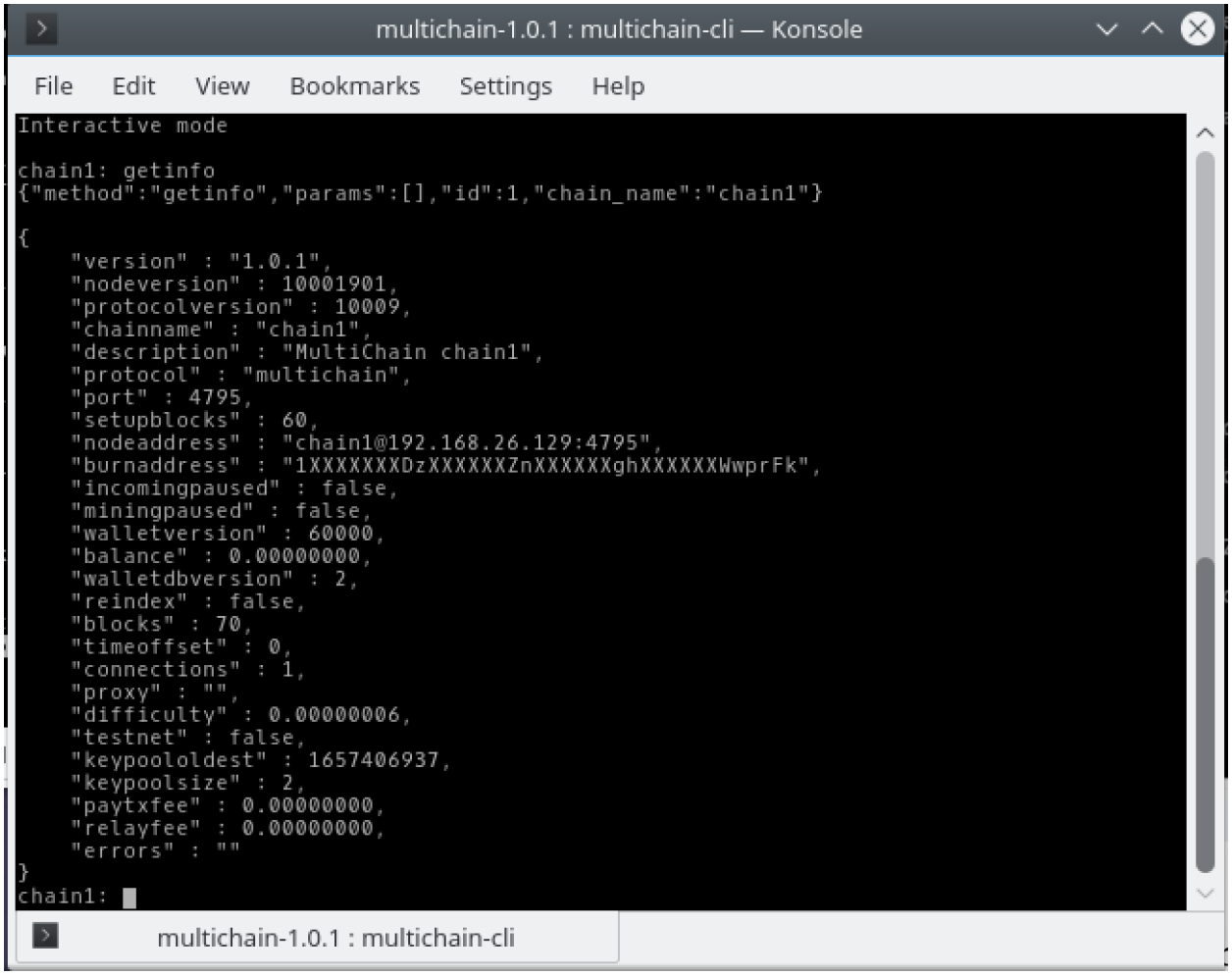
multichaind chain1 –daemon (Server 2)



<u>Step 8:</u> Next, we will proceed with the interactive mode so that we can execute command without adding the multichain-cli chain 1 each and everytime:

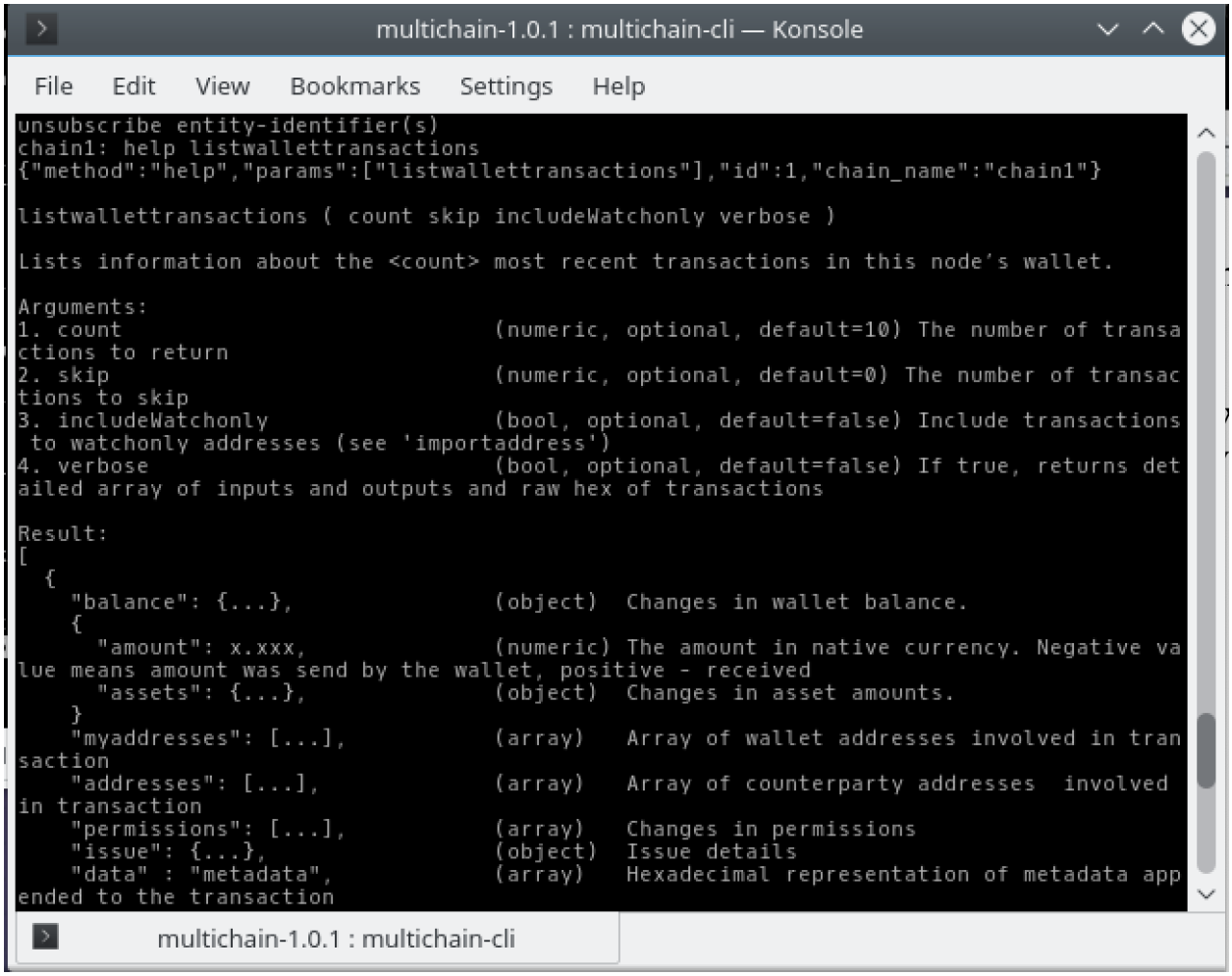
Multichain-cli chain1

You will see the chain1: prompted after entering the above command. Then add **" getinfo "** to get some info regarding the blockchain you are working on.



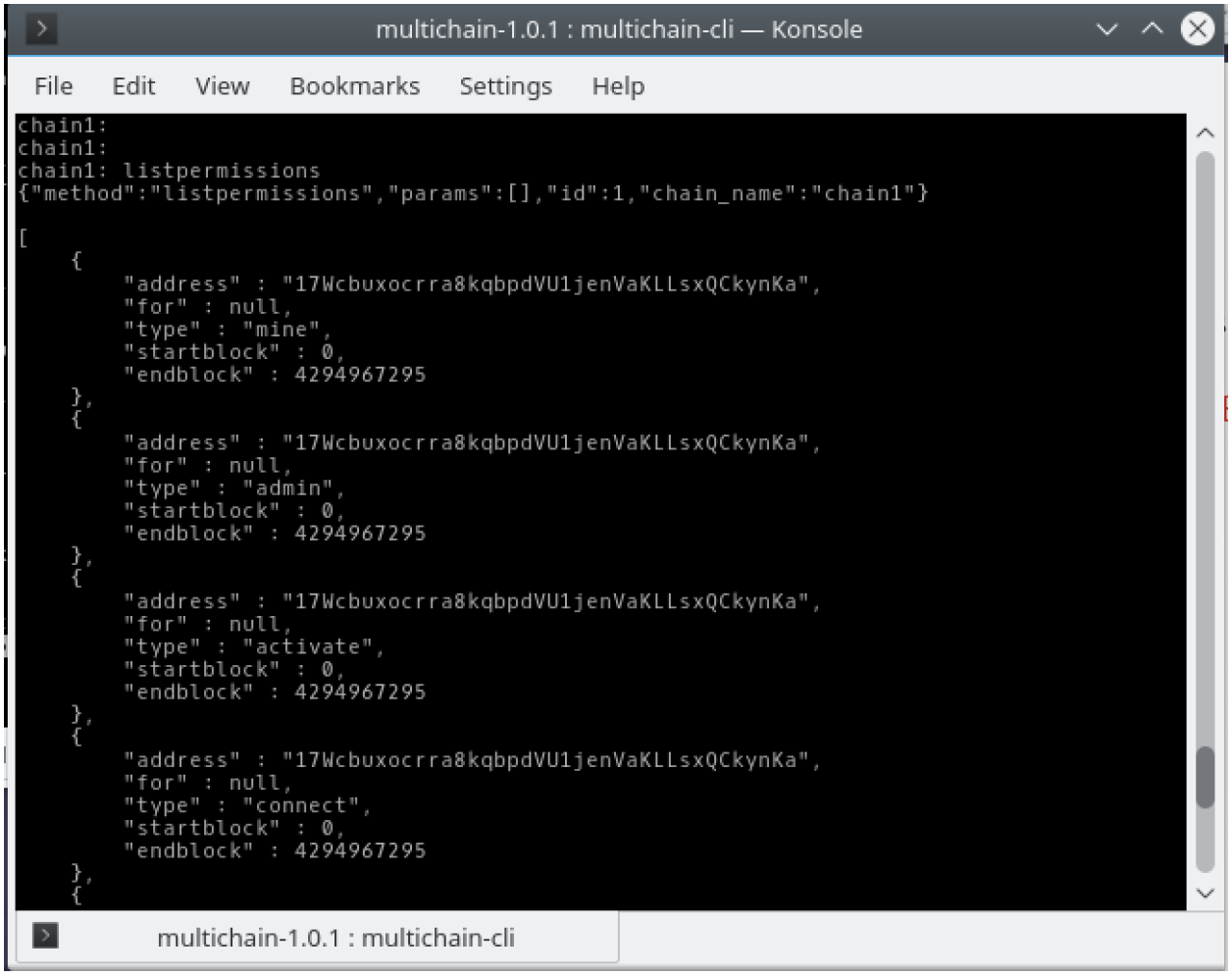
<u>Step 9:</u> Enter " **help** " in the terminal to get all the commands that can be executed in this mode. To see more information about a specific command, enter :

help listwallettransactions



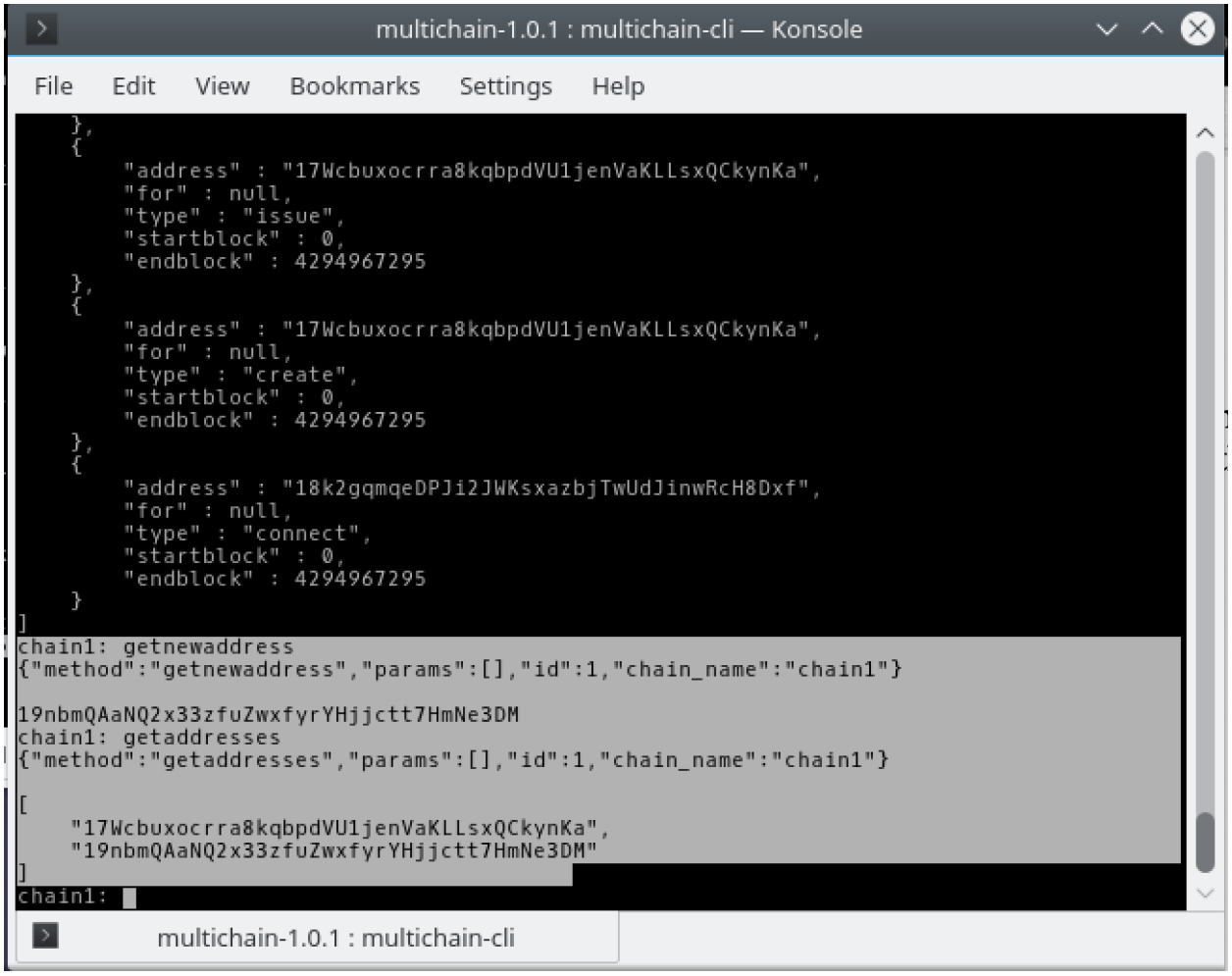
To see all the permission currently assigned to the node,enter :

listpermissions



<u>Step 10:-</u> Now we will create a new address in the wallet which we use to receive the assets.

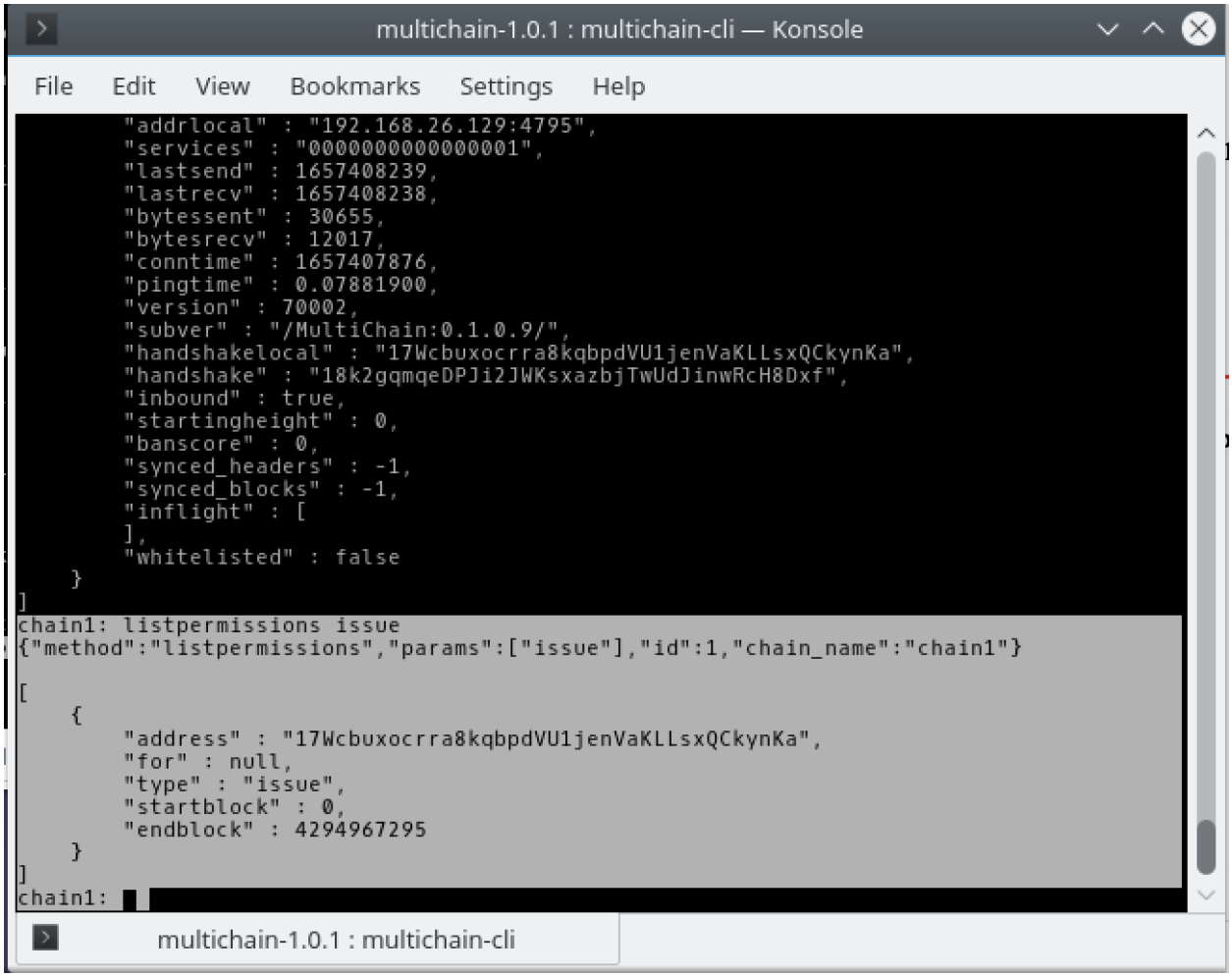
getnewaddress (server 1)



<u>Step 11</u> :- To get a list of all addresses, enter the following command and here you will see two addresses. The first one is the new one revealed by the **"getnewaddress"** command and second one we saw in earlier steps.

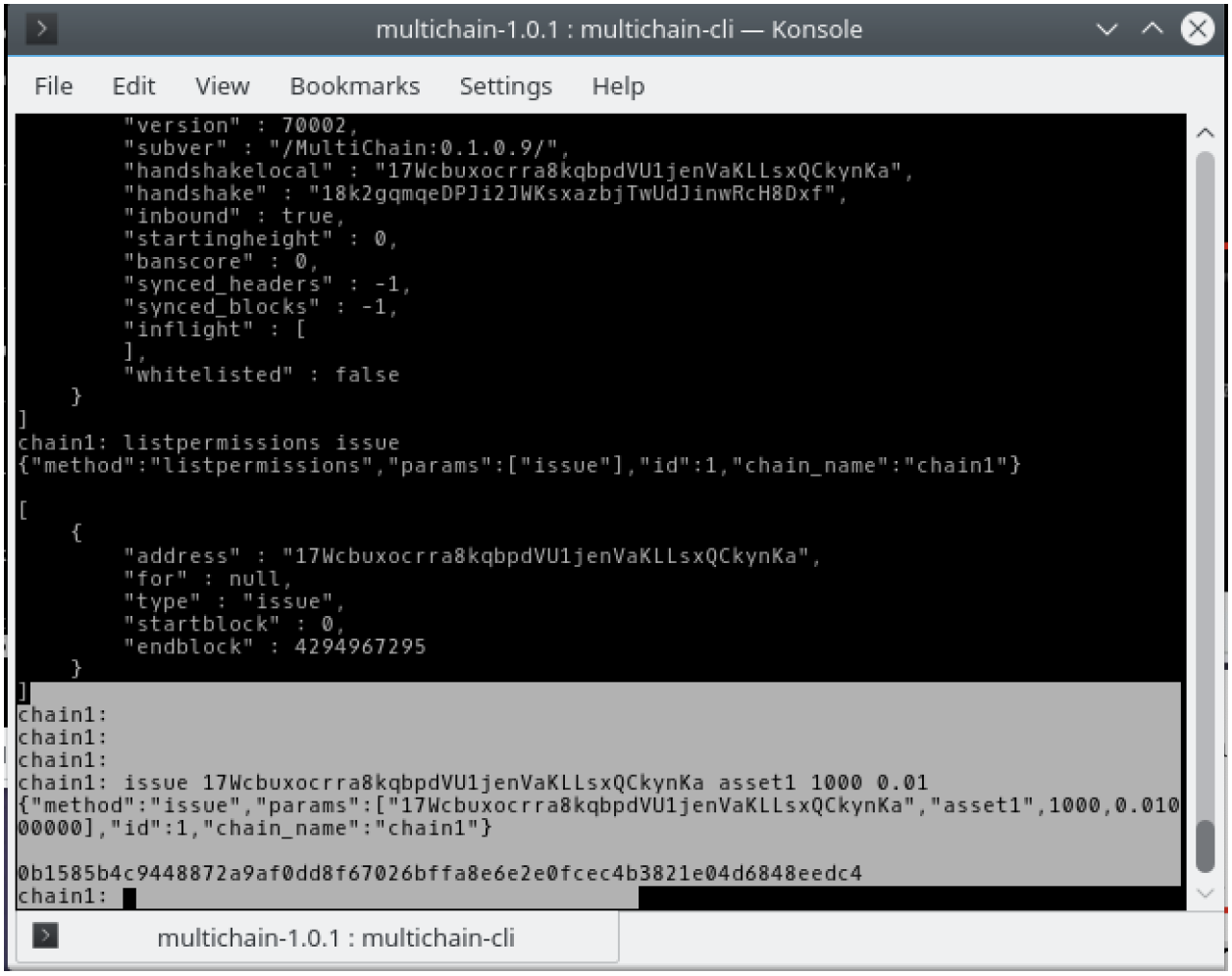
<u>Step 12:-</u> To get the parameters of this blockchain enter **" getblockchainparams "** (server 1).

<u>Step 13:-</u> Now we will create a new asset and send it between nodes (first server) to get a permission enter **" listpermission issue "** to check all the permission. Note the address in NotePad.

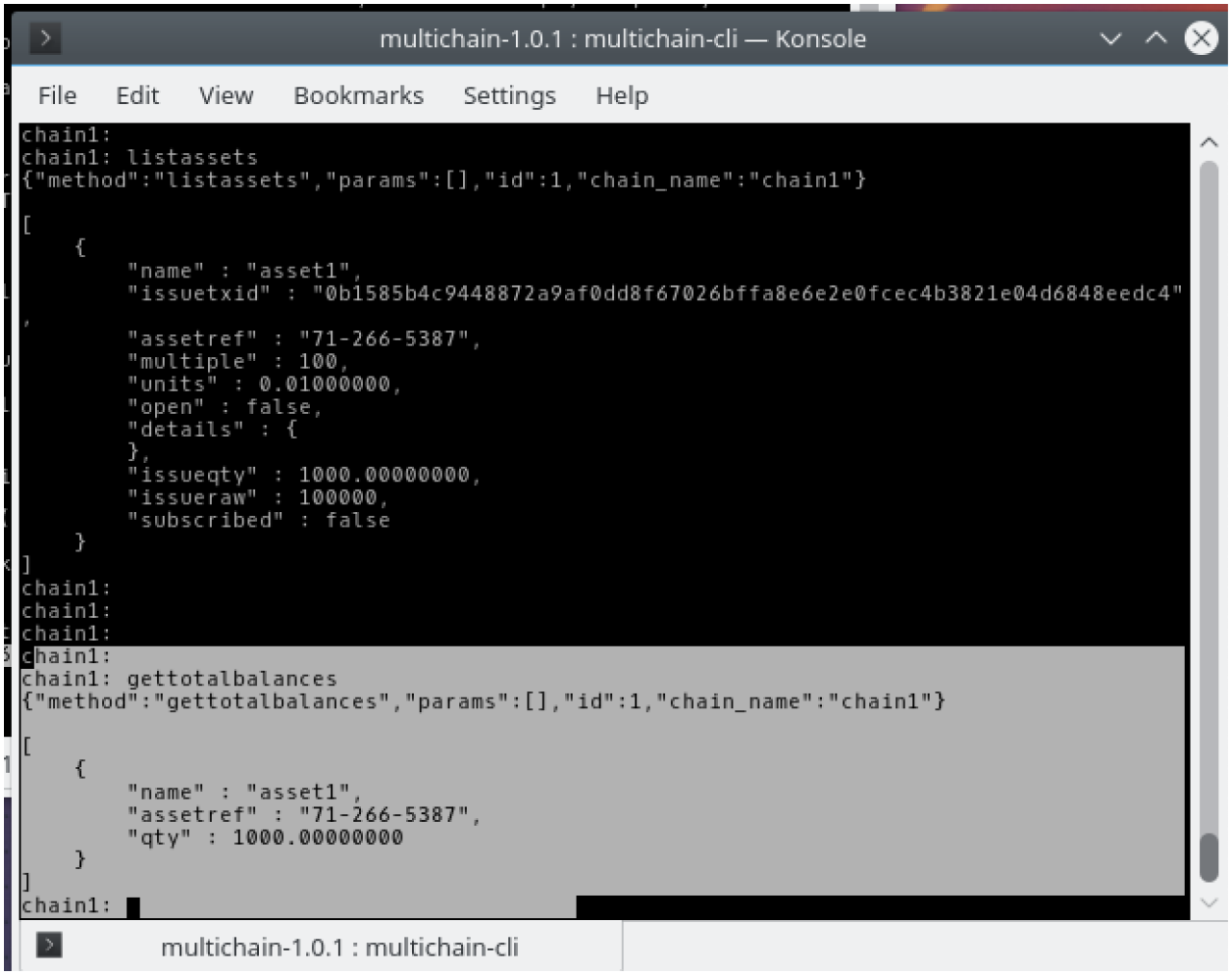


<u>Step 14:-</u> We will issue an asset from the server 1

“ issue *(address you copied in listpermissions command)* assest1 1000 0.01 “ {Server 1}

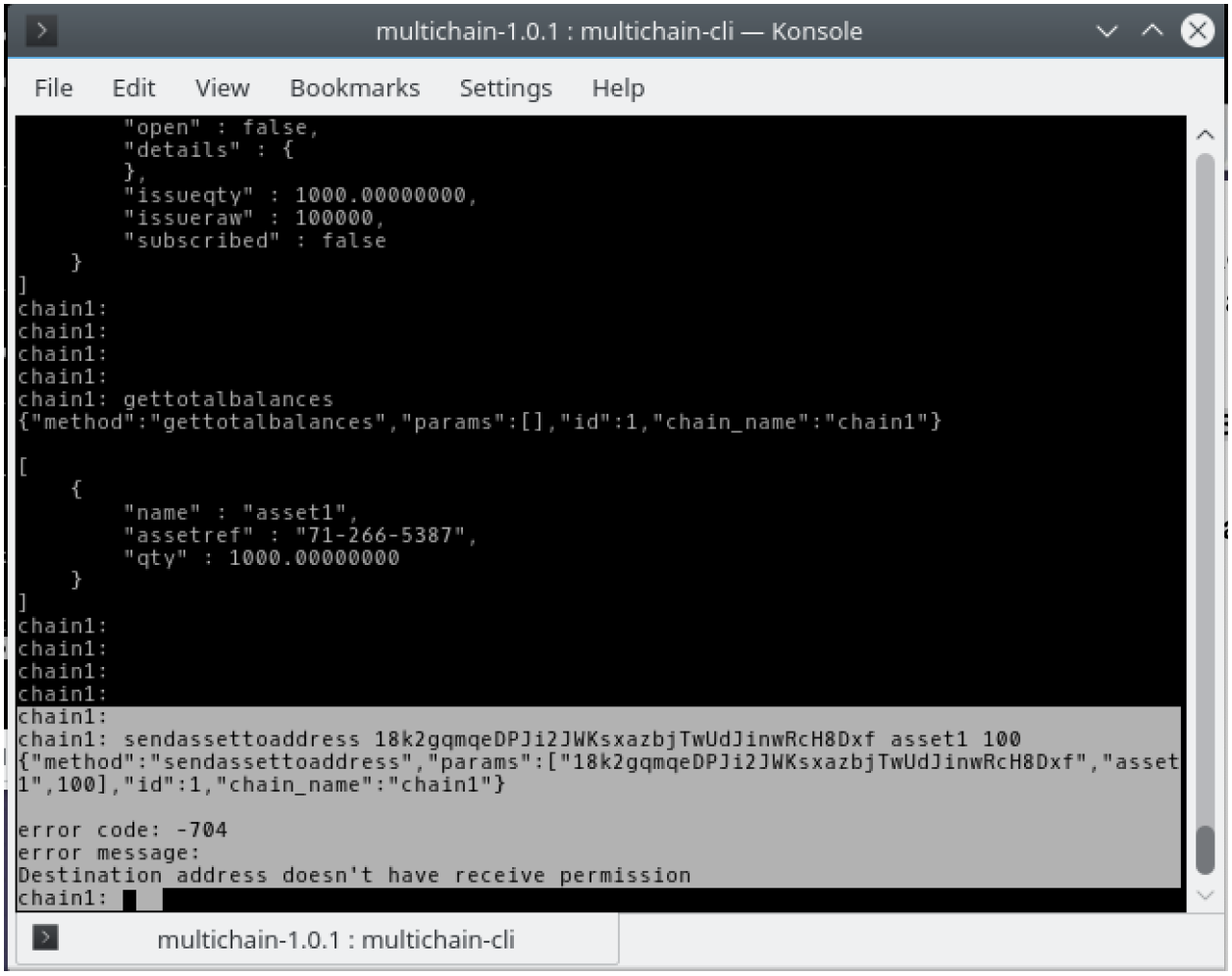


<u>Step 15:-</u> Now we will check our asset on first server **“ gettotalbalances “** first server has 1000 unit of assets. If you check in second server it doesn’t have any asset.



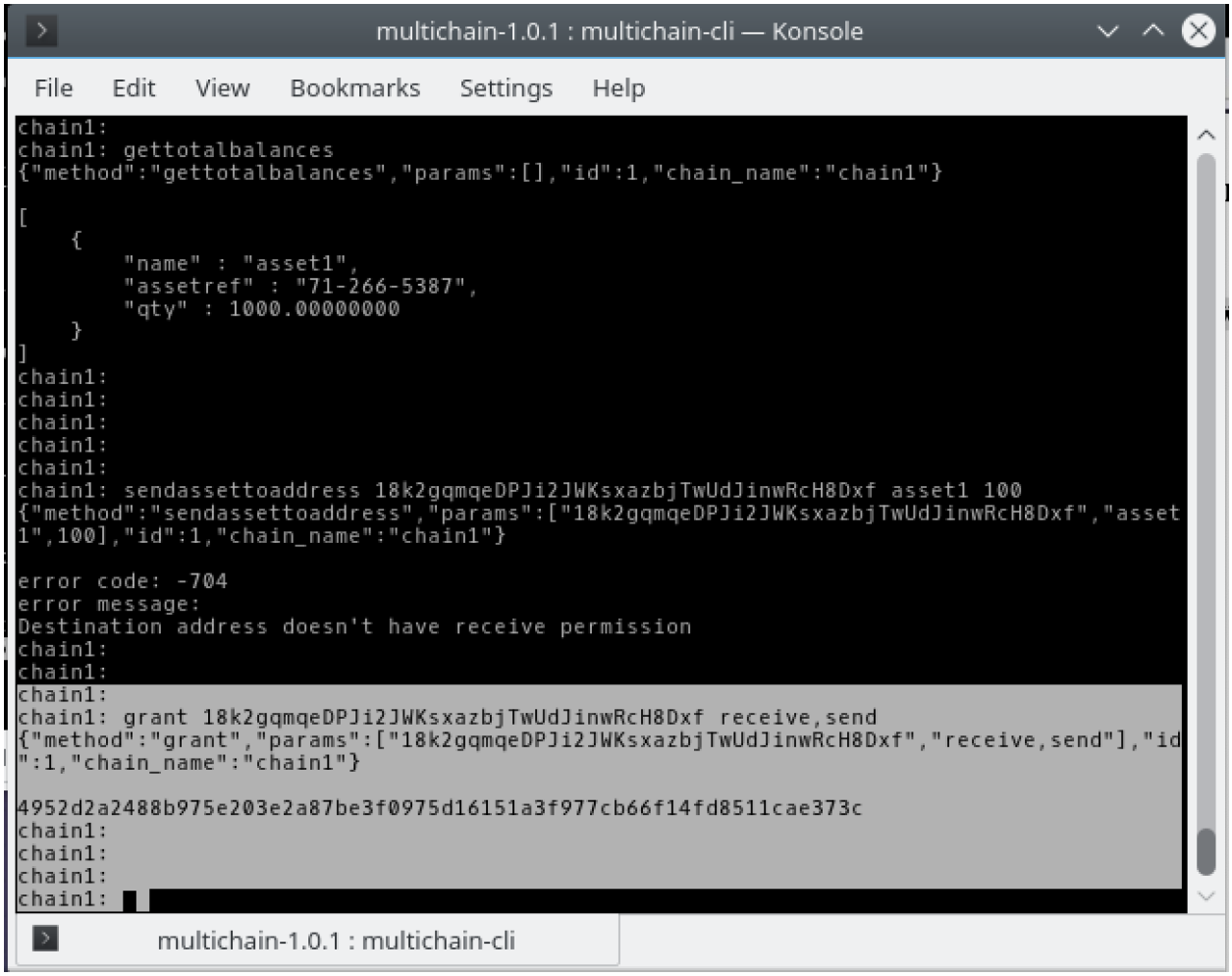
<u>Step 16:-</u> From first server try sending 100 unit of asset to second server wallet

“ sendassettoaddress {here address will be used as earlier copied } asset1 100 “.



<u>Step 17:-</u> Now we will grant, receive and send permission from the first to second server so that the second server can receive an asset from the first unless the server doesn't have permission then no one will be able to send or receive assets.

" **grant {address} receive,send** "



<u>Step 18:-</u> After setting up the permission now we will send the assets to server 2 now we send our assets to server 2 " **sendassettoaddress {address} assets1 100**" from server 1.

<u>Step 19:-</u> On both server check asset balance "gettotalbalances "(server 1 & 2). You can also list a last transaction by entering "**listwallettransactions 1** ". When you hit enter you will see in server one -100 qty that means one server lost 100 assets and server 2 received 100 qty of assets.

multichain-1.0.1 : multichain-cli — Konsole

File Edit View Bookmarks Settings Help

```
error code: -704
error message:
Destination address doesn't have receive permission
chain1:
chain1:
chain1:
chain1: grant 18k2gqmqeDPJi2JWKsxazbjTwUdJinwRcH8Dxf receive,send
{"method":"grant","params":["18k2gqmqeDPJi2JWKsxazbjTwUdJinwRcH8Dxf","receive,send"],"id
":1,"chain_name":"chain1"}

4952d2a2488b975e203e2a87be3f0975d16151a3f977cb66f14fd8511cae373c
chain1:
chain1:
chain1:
chain1: sendassettoaddress 18k2gqmqeDPJi2JWKsxazbjTwUdJinwRcH8Dxf asset1 100
{"method":"sendassettoaddress","params":["18k2gqmqeDPJi2JWKsxazbjTwUdJinwRcH8Dxf","asset
1",100],"id":1,"chain_name":"chain1"}

409f49ead11b23580fc4ffb71272191c444a591eb523c8dc796cb2fceac5faae
chain1:
chain1:
chain1:
chain1:
chain1: gettotalbalances
{"method":"gettotalbalances","params":[],"id":1,"chain_name":"chain1"}

[
    {
        "name" : "asset1",
        "assetref" : "71-266-5387",
        "qty" : 900.00000000
    }
]
chain1:
```

multichain-1.0.1 : multichain-cli

multichain-1.0.1 : multichain-cli — Konsole

File Edit View Bookmarks Settings Help

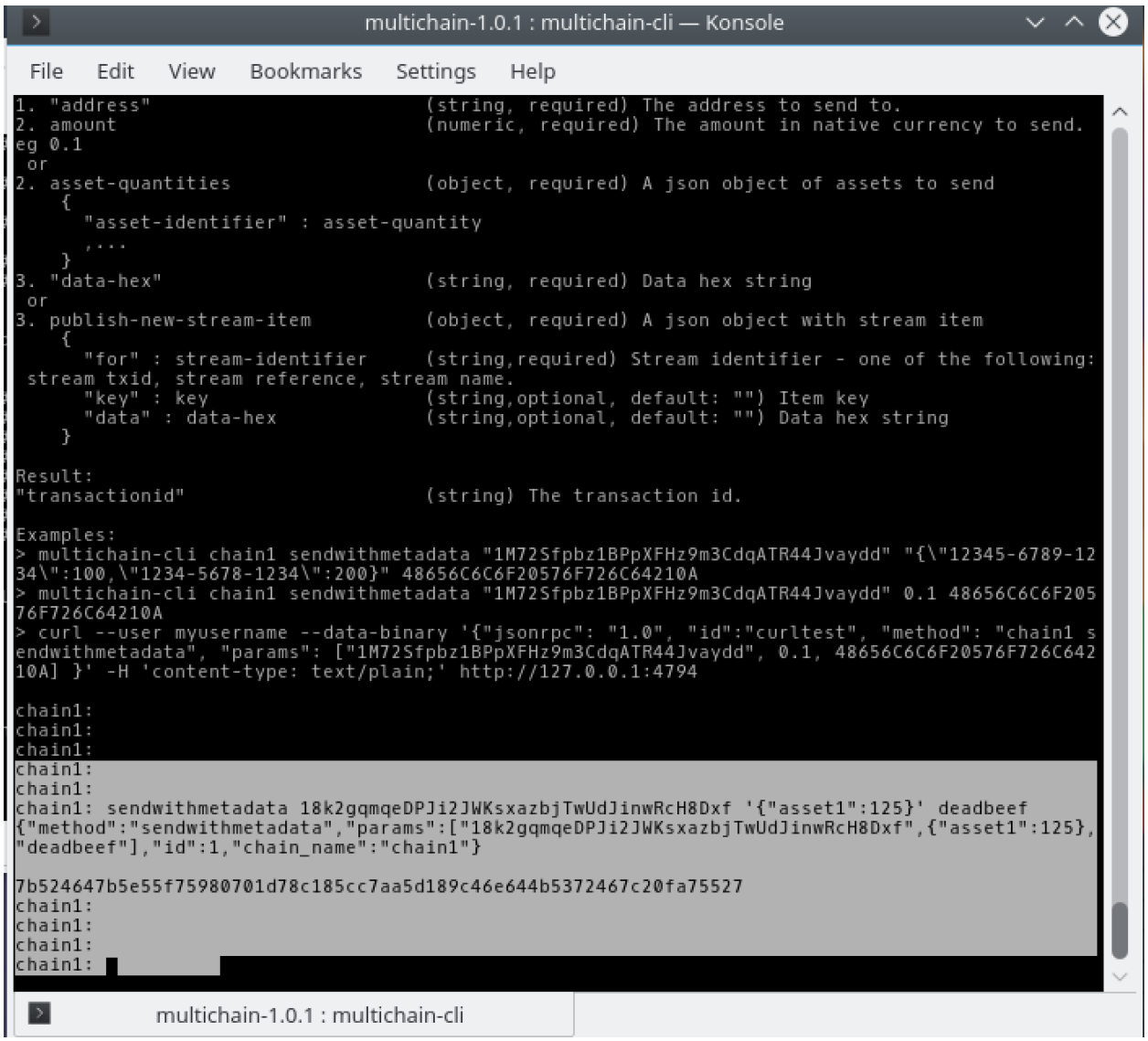
```
        "qty" : 900.00000000
    }
]
chain1:
chain1:
chain1: listwallettransactions 1
{"method":"listwallettransactions","params":[1],"id":1,"chain_name":"chain1"}

[
    {
        "balance" : {
            "amount" : 0.00000000,
            "assets" : [
                {
                    "name" : "asset1",
                    "assetref" : "71-266-5387",
                    "qty" : -100.00000000
                }
            ]
        },
        "myaddresses" : [
            "17Wcbuxocrra8kqbpdVU1jenVaKLLsxQCkynKa"
        ],
        "addresses" : [
            "18k2gqmqeDPJi2JWKsxazbjTwUdJinwRcH8Dxf"
        ],
        "permissions" : [
        ],
        "items" : [
        ],
        "data" : [
        ],
        "confirmations" : 8,
        "blockhash" : "00ea1e3da702d3ca577dd99ea849232f3fc0799a74f6bfe5c890de0de41eb81e"

        "blockindex" : 1,
        "blocktime" : 1657408806,
        "txid" : "409f49ead11b23580fc4ffb71272191c444a591eb523c8dc796cb2fceac5faae",
        "valid" : true,
        "time" : 1657408805,
        "timereceived" : 1657408805
    }
]
chain1:
```

multichain-1.0.1 : multichain-cli

```
multichain-1.0.1 : multichain-cli — Konsole
File  Edit  View  Bookmarks  Settings  Help
        "data" : [
        ],
        "confirmations" : 11,
        "blockhash" : "00ea1e3da702d3ca577dd99ea849232f3fc0799a74f6bfe5c890de0de41eb81e",
        "blockindex" : 1,
        "blocktime" : 1657408806,
        "txid" : "409f49ead11b23580fc4ffb71272191c444a591eb523c8dc796cb2fceac5faae",
        "valid" : true,
        "time" : 1657408805,
        "timereceived" : 1657408805
    }
]
chain1:
chain1:
chain1:
chain1: listwallettransactions 1
{"method":"listwallettransactions","params":[1],"id":1,"chain_name":"chain1"}

[
    {
        "balance" : {
            "amount" : 0.00000000,
            "assets" : [
                {
                    "name" : "asset1",
                    "assetref" : "71-266-5387",
                    "qty" : 100.00000000
                }
            ]
        },
        "myaddresses" : [
            "18k2gqmqeDPJi2JWKsxazbjTwUdJinwRcH8Dxf"
        ],
        "addresses" : [
            "17Wcbuxocrra8kqbpdVU1jenVaKLLsxQCkynKa"
        ],
        "permissions" : [
        ],
        "items" : [
        ],
        "data" : [
        ],
        "confirmations" : 11,
        "blockhash" : "00ea1e3da702d3ca577dd99ea849232f3fc0799a74f6bfe5c890de0de41eb81e",
        "blockindex" : 1,
        "blocktime" : 1657408806,
        "txid" : "409f49ead11b23580fc4ffb71272191c444a591eb523c8dc796cb2fceac5faae",
        "valid" : true,
        "time" : 1657408805,
        "timereceived" : 1657408805
    }
]
chain1:
multichain-1.0.1 : multichain-cli
```

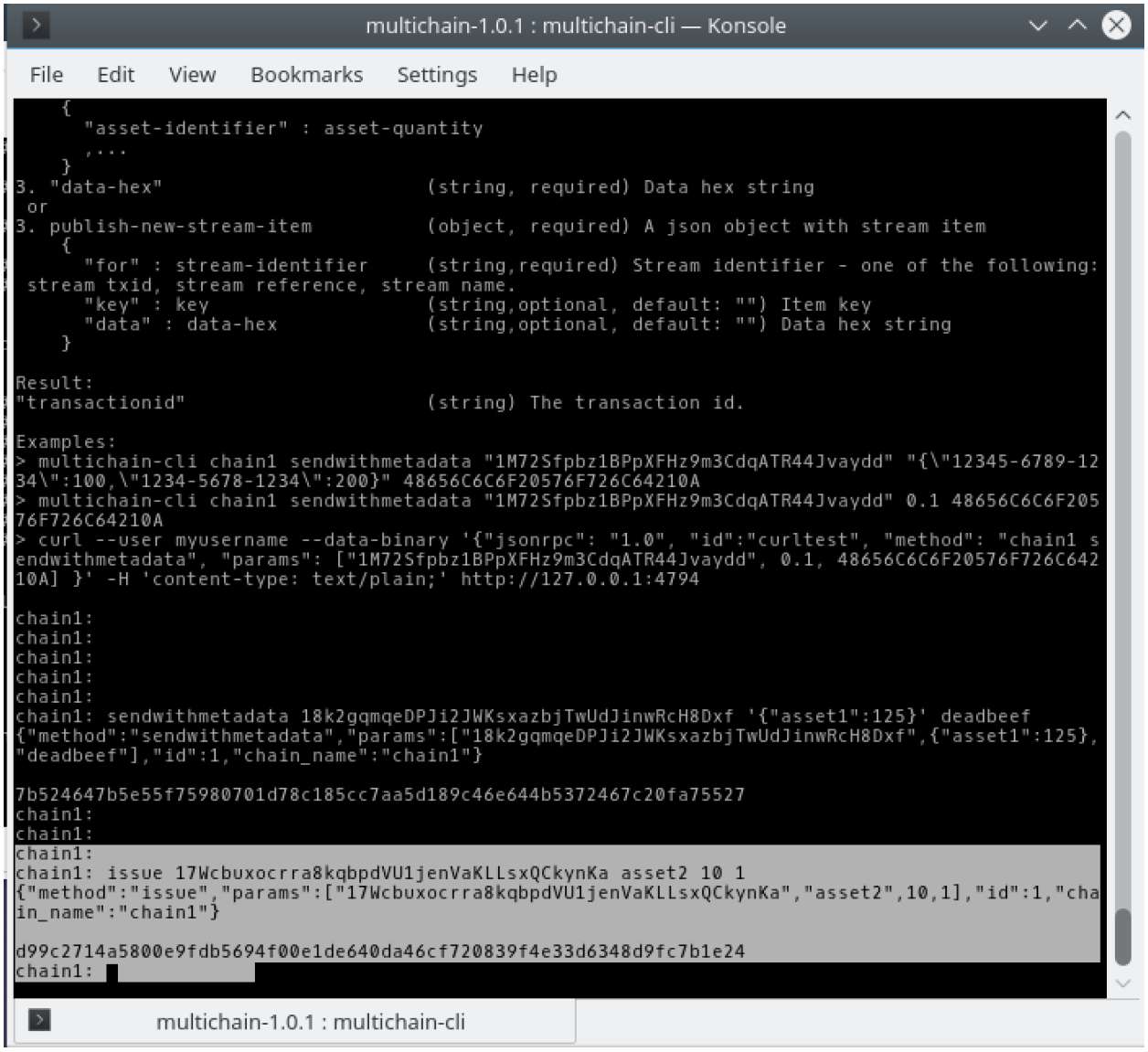
<u>Step 20:-</u> Let's send 125 units of asset1 along with metadata of "deadbeef".

“ **sendwithmetadata ADDRESS '{"asset1":125}' deadbeef** “



<u>Step 21:-</u> Now we'll create a second asset and build an exchange transaction which swaps some of the first asset for some of the second.

"issue ADDRESS asset2 10 1 (Server 1) "



<u>Step 22:-</u> Now check your asset on server 1 **"listassets"** result contains two type of asset 1000 unit of assets1 and 10units of asset 2.

```
multichain-1.0.1 : multichain-cli — Konsole
File  Edit  View  Bookmarks  Settings  Help
"deadbeef"],"id":1,"chain_name":"chain1"}

7b524647b5e55f75980701d78c185cc7aa5d189c46e644b5372467c20fa75527
chain1:
chain1:
chain1:
chain1: issue 17Wcbuxocrra8kqbpdVU1jenVaKLLsxQCkynKa asset2 10 1
{"method":"issue","params":["17Wcbuxocrra8kqbpdVU1jenVaKLLsxQCkynKa","asset2",10,1],"id":1,"cha
in_name":"chain1"}

d99c2714a5800e9fdb5694f00e1de640da46cf720839f4e33d6348d9fc7b1e24
chain1:
chain1:
chain1: listassets
{"method":"listassets","params":[],"id":1,"chain_name":"chain1"}

[
    {
        "name" : "asset1",
        "issuetxid" : "0b1585b4c9448872a9af0dd8f67026bffa8e6e2e0fcec4b3821e04d6848eedc4",
        "assetref" : "71-266-5387",
        "multiple" : 100,
        "units" : 0.01000000,
        "open" : false,
        "details" : {
        },
        "issueqty" : 1000.00000000,
        "issueraw" : 100000,
        "subscribed" : false
    },
    {
        "name" : "asset2",
        "issuetxid" : "d99c2714a5800e9fdb5694f00e1de640da46cf720839f4e33d6348d9fc7b1e24",
        "assetref" : "112-266-40153",
        "multiple" : 1,
        "units" : 1.00000000,
        "open" : false,
        "details" : {
        },
        "issueqty" : 10.00000000,
        "issueraw" : 10,
        "subscribed" : false
    }
]
chain1:
multichain-1.0.1 : multichain-cli
```

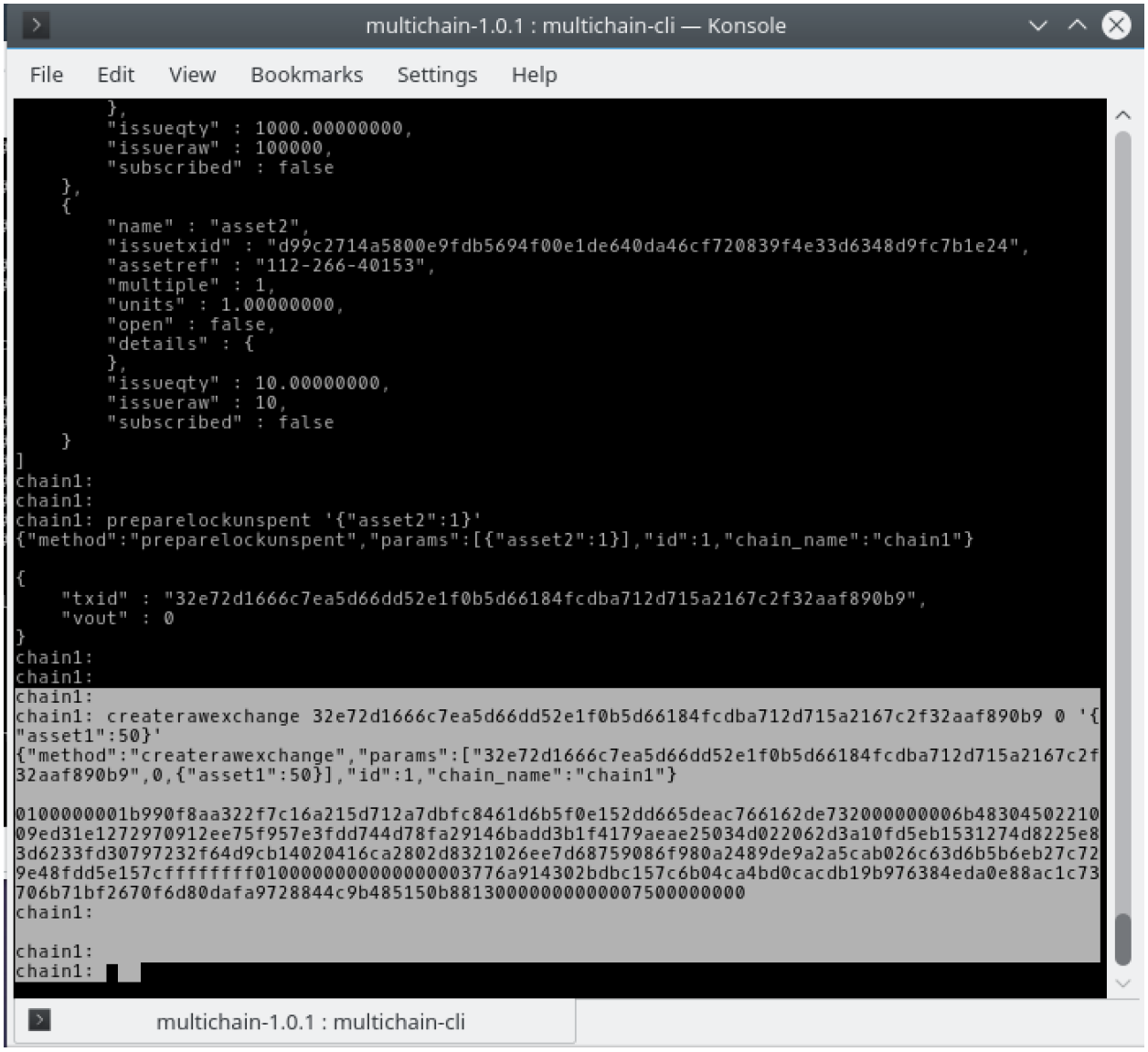
<u>Step 23:-</u> Now we are going to swap 50unit of asset1 for 1unit of asset2.

" **preparelockunspent '{"asset2 : 1"}'** "

```
multichain-1.0.1 : multichain-cli — Konsole
File   Edit   View   Bookmarks   Settings   Help

d99c2714a5800e9fdb5694f00e1de640da46cf720839f4e33d6348d9fc7b1e24
chain1:
chain1:
chain1: listassets
{"method":"listassets","params":[],"id":1,"chain_name":"chain1"}

[
    {
        "name" : "asset1",
        "issuetxid" : "0b1585b4c9448872a9af0dd8f67026bffa8e6e2e0fcec4b3821e04d6848eedc4",
        "assetref" : "71-266-5387",
        "multiple" : 100,
        "units" : 0.01000000,
        "open" : false,
        "details" : {
        },
        "issueqty" : 1000.00000000,
        "issueraw" : 100000,
        "subscribed" : false
    },
    {
        "name" : "asset2",
        "issuetxid" : "d99c2714a5800e9fdb5694f00e1de640da46cf720839f4e33d6348d9fc7b1e24",
        "assetref" : "112-266-40153",
        "multiple" : 1,
        "units" : 1.00000000,
        "open" : false,
        "details" : {
        },
        "issueqty" : 10.00000000,
        "issueraw" : 10,
        "subscribed" : false
    }
]
chain1:
chain1:
chain1: preparelockunspent '{"asset2":1}'
{"method":"preparelockunspent","params":[{"asset2":1}],"id":1,"chain_name":"chain1"}

{
    "txid" : "32e72d1666c7ea5d66dd52e1f0b5d66184fcdba712d715a2167c2f32aaf890b9",
    "vout" : 0
}
chain1:
multichain-1.0.1 : multichain-cli
```

<u>Step 24:-</u> Execute below command, using the txid you got from the previous command.

createrawexchange *"TXID"* **0 '{"asset1":50}'** (*in server 1)*



After executing the above command you will get a hexadecimal copy that in notepad.

decoderawexchange *[paste-hex-blob]* **(Server 1)**

multichain-1.0.1 : multichain-cli — Konsole

File Edit View Bookmarks Settings Help

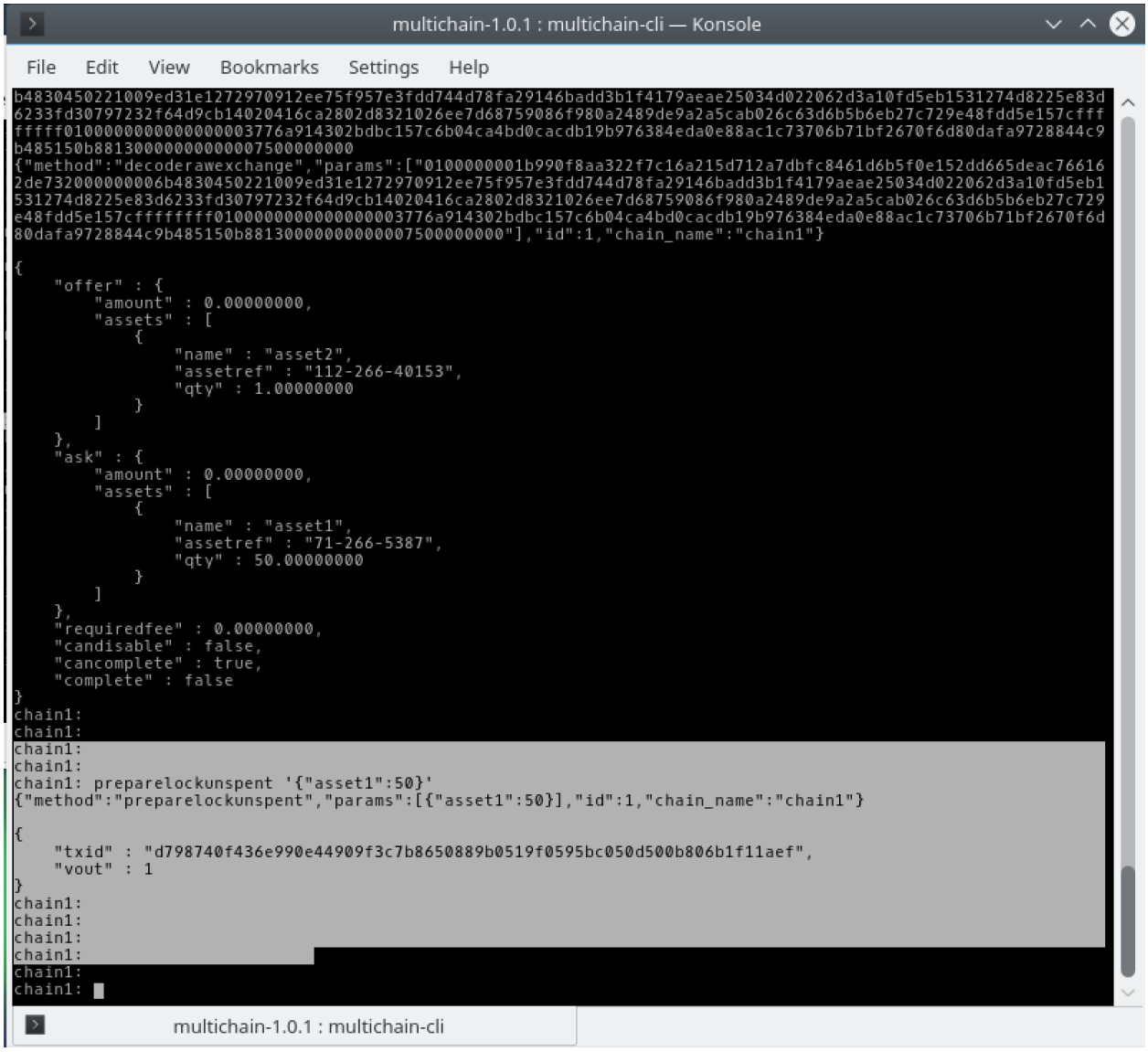
```
    "blockindex" : 1,
    "blocktime" : 1657409255,
    "txid" : "7b524647b5e55f75980701d78c185cc7aa5d189c46e644b5372467c20fa75527",
    "valid" : true,
    "time" : 1657409240,
    "timereceived" : 1657409240
}
chain1:
chain1:
chain1:
chain1:
chain1:
chain1: decoderawexchange 0100000001b990f8aa322f7c16a215d712a7dbfc8461d6b5f0e152dd665deac766162de732000000006
b4830450221009ed31e1272970912ee75f957e3fdd744d78fa29146badd3b1f4179aeae25034d022062d3a10fd5eb1531274d8225e83d
6233fd30797232f64d9cb14020416ca2802d8321026ee7d68759086f980a2489de9a2a5cab026c63d6b5b6eb27c729e48fdd5e157cfff
fffff01000000000000000003776a914302bdbc157c6b04ca4bd0cacdb19b976384eda0e88ac1c73706b71bf2670f6d80dafa9728844c9
b485150b88130000000000000007500000000
{"method":"decoderawexchange","params":["0100000001b990f8aa322f7c16a215d712a7dbfc8461d6b5f0e152dd665deac76616
2de732000000006b4830450221009ed31e1272970912ee75f957e3fdd744d78fa29146badd3b1f4179aeae25034d022062d3a10fd5eb1
531274d8225e83d6233fd30797232f64d9cb14020416ca2802d8321026ee7d68759086f980a2489de9a2a5cab026c63d6b5b6eb27c729
e48fdd5e157cffffffff01000000000000000003776a914302bdbc157c6b04ca4bd0cacdb19b976384eda0e88ac1c73706b71bf2670f6d
80dafa9728844c9b485150b88130000000000000007500000000"],"id":1,"chain_name":"chain1"}

{
    "offer" : {
        "amount" : 0.00000000,
        "assets" : [
            {
                "name" : "asset2",
                "assetref" : "112-266-40153",
                "qty" : 1.00000000
            }
        ]
    },
    "ask" : {
        "amount" : 0.00000000,
        "assets" : [
            {
                "name" : "asset1",
                "assetref" : "71-266-5387",
                "qty" : 50.00000000
            }
        ]
    },
    "requiredfee" : 0.00000000,
    "candisable" : false,
    "cancomplete" : true,
    "complete" : false
}
chain1:
chain1:
chain1:
chain1:
```

multichain-1.0.1 : multichain-cli

<u>Step 25:-</u> On the 2nd server, create a transaction output containing 50unit of asset .

preparelockunspent '{"asset1":50}' (Server 2)

Note : The txid and vout values returned.



<u>Step 26:-</u> To prepare the exchange transaction execute:

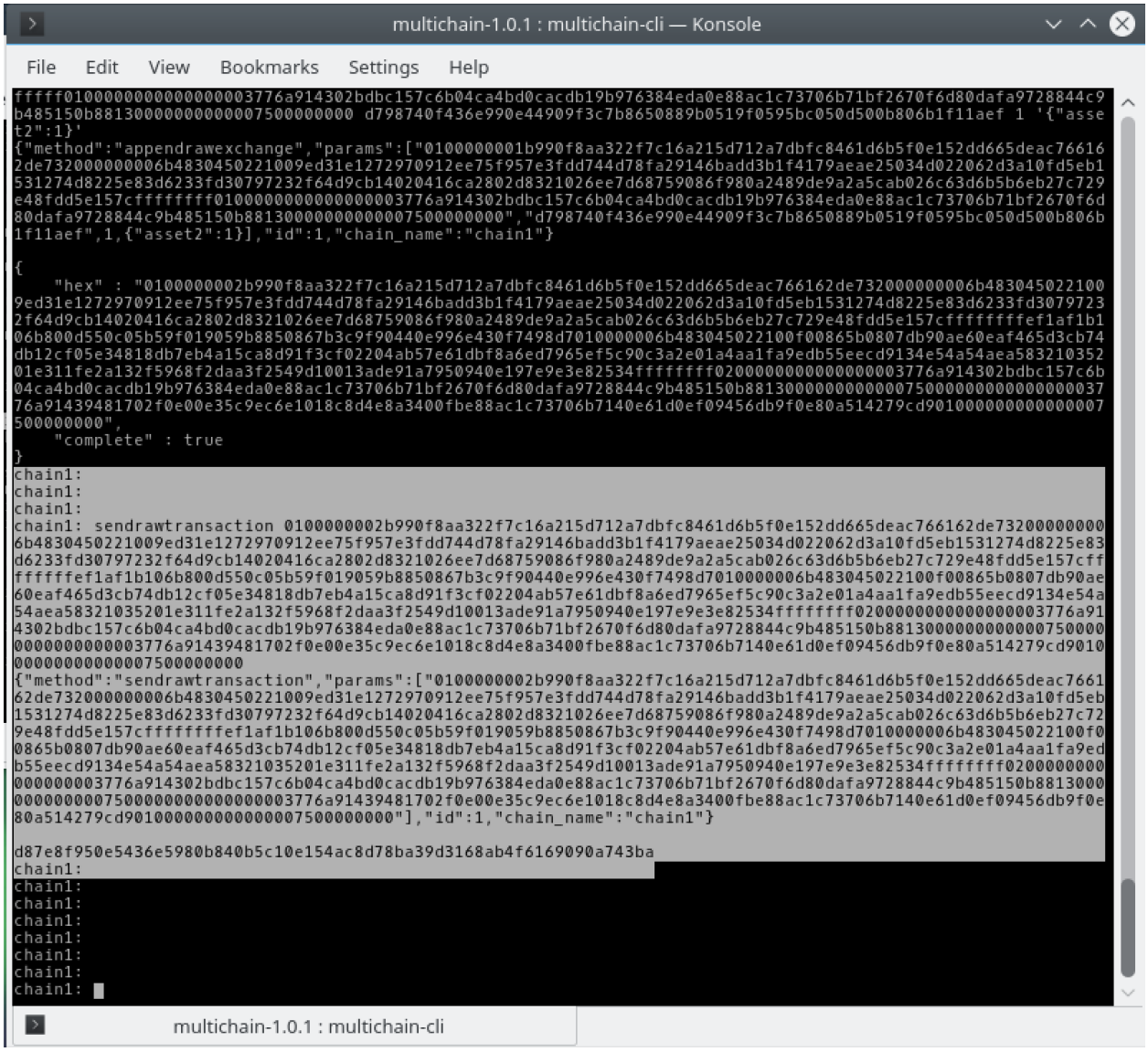
appendrawexchange [paste-hex-blob] [txid] '{"asset2":1} (Server 2)

The output should contain true after the long hexadecimal box.

```
multichain-1.0.1 : multichain-cli — Konsole
File   Edit   View   Bookmarks   Settings   Help
                "name" : "asset1",
                "assetref" : "71-266-5387",
                "qty" : 50.00000000
            }
        ]
    },
    "requiredfee" : 0.00000000,
    "candisable" : false,
    "cancomplete" : true,
    "complete" : false
}
chain1:
chain1:
chain1:
chain1:
chain1: preparelockunspent '{"asset1":50}'
{"method":"preparelockunspent","params":[{"asset1":50}],"id":1,"chain_name":"chain1"}

{
    "txid" : "d798740f436e990e44909f3c7b8650889b0519f0595bc050d500b806b1f11aef",
    "vout" : 1
}
chain1:
chain1:
chain1:
chain1:
chain1:
chain1: appendrawexchange 0100000001b990f8aa322f7c16a215d712a7dbfc8461d6b5f0e152dd665deac766162de732000000006
b4830450221009ed31e1272970912ee75f957e3fdd744d78fa29146badd3b1f4179aeae25034d022062d3a10fd5eb1531274d8225e83d
6233fd30797232f64d9cb14020416ca2802d8321026ee7d68759086f980a2489de9a2a5cab026c63d6b5b6eb27c729e48fdd5e157cfff
ffffff0100000000000000000003776a914302bdbc157c6b04ca4bd0cacdb19b976384eda0e88ac1c73706b71bf2670f6d80dafa9728844c9
b485150b88130000000000000007500000000 d798740f436e990e44909f3c7b8650889b0519f0595bc050d500b806b1f11aef 1 '{"asse
t2":1}'
{"method":"appendrawexchange","params":["0100000001b990f8aa322f7c16a215d712a7dbfc8461d6b5f0e152dd665deac76616
2de732000000006b4830450221009ed31e1272970912ee75f957e3fdd744d78fa29146badd3b1f4179aeae25034d022062d3a10fd5eb1
531274d8225e83d6233fd30797232f64d9cb14020416ca2802d8321026ee7d68759086f980a2489de9a2a5cab026c63d6b5b6eb27c729
e48fdd5e157cffffffff0100000000000000000003776a914302bdbc157c6b04ca4bd0cacdb19b976384eda0e88ac1c73706b71bf2670f6d
80dafa9728844c9b485150b88130000000000000007500000000","d798740f436e990e44909f3c7b8650889b0519f0595bc050d500b806b
1f11aef",1,{"asset2":1}],"id":1,"chain_name":"chain1"}

{
    "hex" : "0100000002b990f8aa322f7c16a215d712a7dbfc8461d6b5f0e152dd665deac766162de732000000006b483045022100
9ed31e1272970912ee75f957e3fdd744d78fa29146badd3b1f4179aeae25034d022062d3a10fd5eb1531274d8225e83d6233fd3079723
2f64d9cb14020416ca2802d8321026ee7d68759086f980a2489de9a2a5cab026c63d6b5b6eb27c729e48fdd5e157cffffffffef1af1b1
06b800d550c05b59f019059b8850867b3c9f90440e996e430f7498d7010000006b483045022100f00865b0807db90ae60eaf465d3cb74
db12cf05e34818db7eb4a15ca8d91f3cf02204ab57e61dbf8a6ed7965ef5c90c3a2e01a4aa1fa9edb55eecd9134e54a54aea583210352
01e311fe2a132f5968f2daa3f2549d10013ade91a7950940e197e9e3e82534ffffffff0200000000000000003776a914302bdbc157c6b
04ca4bd0cacdb19b976384eda0e88ac1c73706b71bf2670f6d80dafa9728844c9b485150b88130000000000000007500000000000000003
76a91439481702f0e00e35c9ec6e1018c8d4e8a3400fbe88ac1c73706b7140e61d0ef09456db9f0e80a514279cd90100000000000000007
500000000",
    "complete" : true
}
chain1:
multichain-1.0.1 : multichain-cli
```

<u>Step 27:-</u> This final hex blob is a raw transaction representing the completed exchange.

Sendrawtransaction [paste -longer hex blob] (server 2)



To verify that the exchange was successful, execute
gettotalbalances 0 (server 1) & (server 2)

Proj 9 Multichain 1

multichain-1.0.1 : multichain-cli — Konsole

File Edit View Bookmarks Settings Help

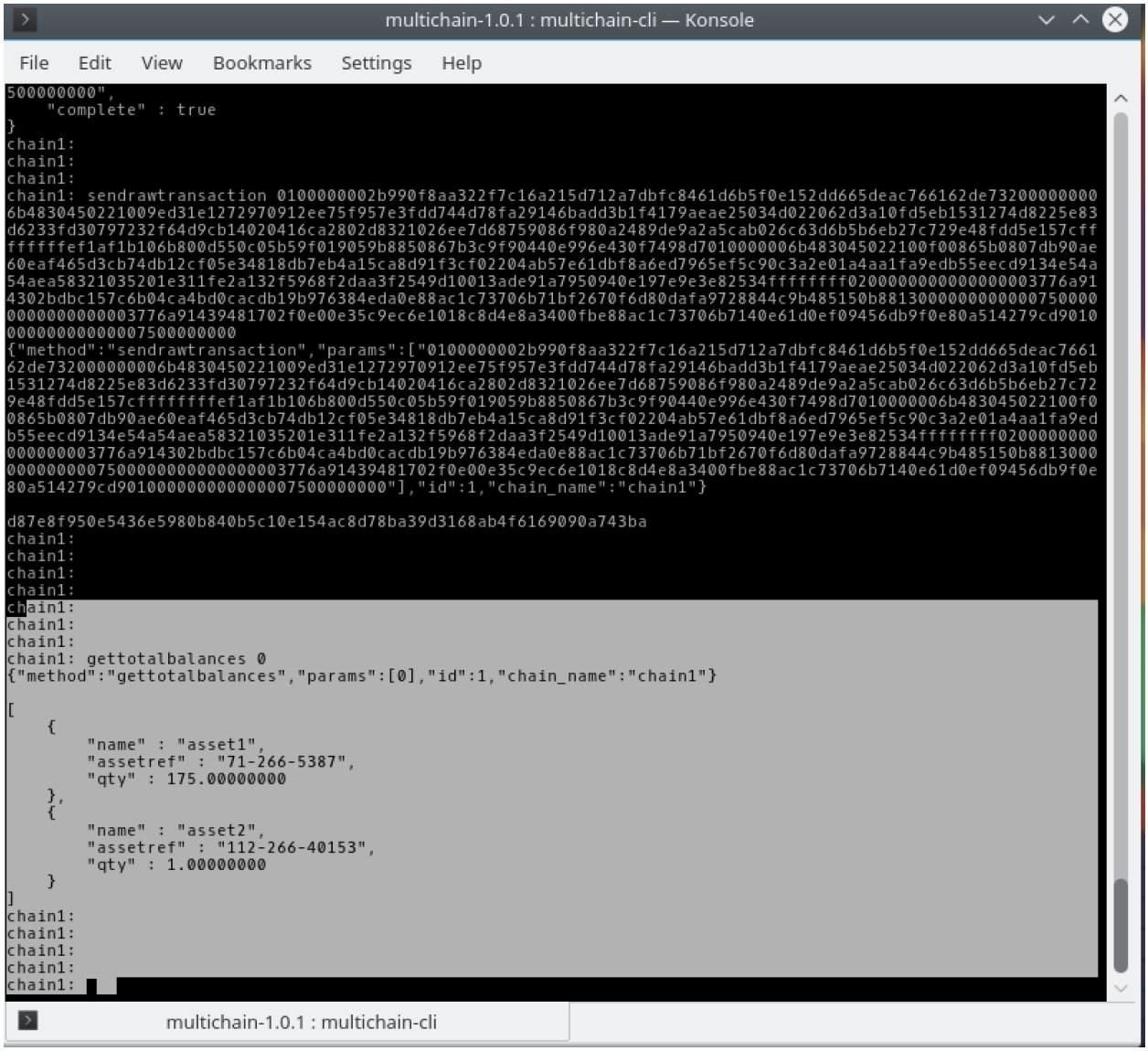
```
{"method":"preparelockunspent","params":[{"asset2":1}],"id":1,"chain_name":"chain1"}

{
    "txid" : "32e72d1666c7ea5d66dd52e1f0b5d66184fcdba712d715a2167c2f32aaf890b9",
    "vout" : 0
}
chain1:
chain1:
chain1:
chain1: createrawexchange 32e72d1666c7ea5d66dd52e1f0b5d66184fcdba712d715a2167c2f32aaf890b9 0 '{
"asset1":50}'
{"method":"createrawexchange","params":["32e72d1666c7ea5d66dd52e1f0b5d66184fcdba712d715a2167c2f
32aaf890b9",0,{"asset1":50}],"id":1,"chain_name":"chain1"}

0100000001b990f8aa322f7c16a215d712a7dbfc8461d6b5f0e152dd665deac766162de732000000006b48304502210
09ed31e1272970912ee75f957e3fdd744d78fa29146badd3b1f4179aeae25034d022062d3a10fd5eb1531274d8225e8
3d6233fd30797232f64d9cb14020416ca2802d8321026ee7d68759086f980a2489de9a2a5cab026c63d6b5b6eb27c72
9e48fdd5e157cffffffff01000000000000000003776a914302bdbc157c6b04ca4bd0cacdb19b976384eda0e88ac1c73
706b71bf2670f6d80dafa9728844c9b485150b8813000000000000007500000000
chain1:

chain1:
chain1:
chain1:
chain1:
chain1: gettotalbalances 0



{"method":"gettotalbalances","params":[0],"id":1,"chain_name":"chain1"}

[
    {
        "name" : "asset2",
        "assetref" : "112-266-40153",
        "qty" : 9.00000000
    },
    {
        "name" : "asset1",
        "assetref" : "71-266-5387",
        "qty" : 825.00000000
    }
]
chain1:
```

multichain-1.0.1 : multichain-cli

multichain-1.0.1 : multichain-cli — K

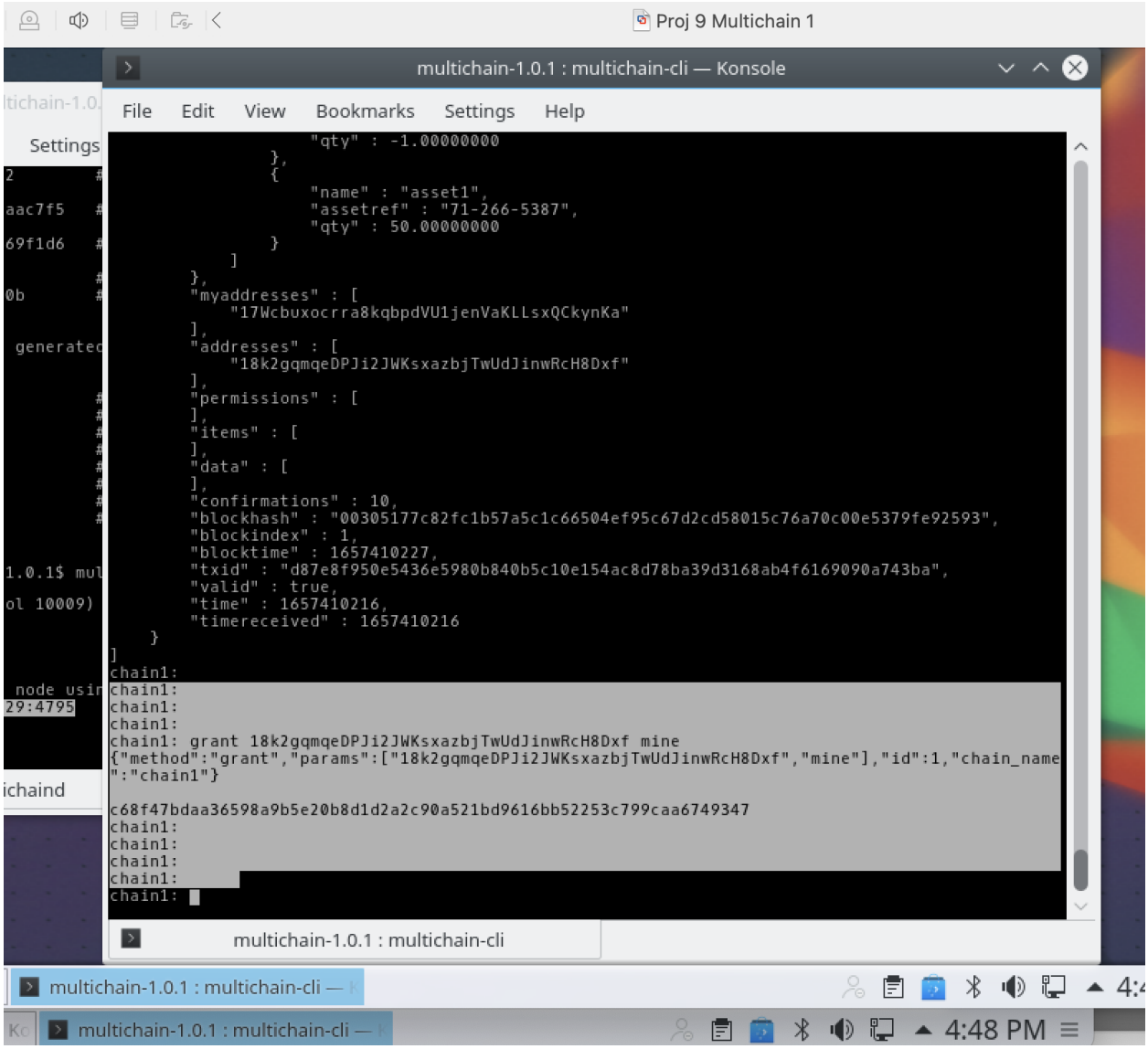
multichain-1.0.1 : multichain-cli — K

4:44 PM



<u>Step 28:-</u> Now we will start Round-robin mining between the nodes. To do that we need to give the permission to the second server to mine. On the 1st server execute:

grant [server2 address] mine (server 1)



<u>Step 29:-</u> On the second server execute this command to see weather 2nd server got the permission to mine:

listpermissions mine (server 2)

```
multichain-1.0.1 : multichain-cli — Konsole
File  Edit  View  Bookmarks  Settings  Help
        "blockhash" : "00305177c82fc1b57a5c1c66504ef95c67d2cd58015c76a70c00e5379fe92593",
        "blockindex" : 1,
        "blocktime" : 1657410227,
        "txid" : "d87e8f950e5436e5980b840b5c10e154ac8d78ba39d3168ab4f6169090a743ba",
        "valid" : true,
        "time" : 1657410216,
        "timereceived" : 1657410216
    }
]
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1: listpermissions mine
{"method":"listpermissions","params":["mine"],"id":1,"chain_name":"chain1"}

[
    {
        "address" : "17Wcbuxocrra8kqbpdVU1jenVaKLLsxQCkynKa",
        "for" : null,
        "type" : "mine",
        "startblock" : 0,
        "endblock" : 4294967295
    },
    {
        "address" : "18k2gqmqeDPJi2JWKsxazbjTwUdJinwRcH8Dxf",
        "for" : null,
        "type" : "mine",
        "startblock" : 0,
        "endblock" : 4294967295
    }
]
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
multichain-1.0.1 : multichain-cli
```

Now wait for couple of minute to mine few blocks, then on either one of the server execute to check the current block height:

getinfo

multichain-1.0.1 : multichain-cli — Konsole

File Edit View Bookmarks Settings Help

```
]
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1: getinfo
{"method":"getinfo","params":[],"id":1,"chain_name":"chain1"}

{
    "version" : "1.0.1",
    "nodeversion" : 10001901,
    "protocolversion" : 10009,
    "chainname" : "chain1",
    "description" : "MultiChain chain1",
    "protocol" : "multichain",
    "port" : 4795,
    "setupblocks" : 60,
    "nodeaddress" : "chain1@192.168.26.130:4795",
    "burnaddress" : "1XXXXXXXDzXXXXXXZnXXXXXXghXXXXXXWwprFk",
    "incomingpaused" : false,
    "miningpaused" : false,
    "walletversion" : 60000,
    "balance" : 0.00000000,
    "walletdbversion" : 2,
    "reindex" : false,
    "blocks" : 161,
    "timeoffset" : 0,
    "connections" : 1,
    "proxy" : "",
    "difficulty" : 0.00000006,
    "testnet" : false,
    "keypoololdest" : 1657407652,
    "keypoolsize" : 2,
    "paytxfee" : 0.00000000,
    "relayfee" : 0.00000000,
    "errors" : ""
}
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
```

multichain-1.0.1 : multichain-cli

Now to get the hashes of last few blocks:

getblocks (enter number of block you see)

multichain-1.0.1 : multichain-cli — Konsole

File Edit View Bookmarks Settings Help

```
chain1:
chain1: getinfo
{"method":"getinfo","params":[],"id":1,"chain_name":"chain1"}

{
    "version" : "1.0.1",
    "nodeversion" : 10001901,
    "protocolversion" : 10009,
    "chainname" : "chain1",
    "description" : "MultiChain chain1",
    "protocol" : "multichain",
    "port" : 4795,
    "setupblocks" : 60,
    "nodeaddress" : "chain1@192.168.26.130:4795",
    "burnaddress" : "1XXXXXXXDzXXXXXXZnXXXXXXghXXXXXXWwprFk",
    "incomingpaused" : false,
    "miningpaused" : false,
    "walletversion" : 60000,
    "balance" : 0.00000000,
    "walletdbversion" : 2,
    "reindex" : false,
    "blocks" : 161,
    "timeoffset" : 0,
    "connections" : 1,
    "proxy" : "",
    "difficulty" : 0.00000006,
    "testnet" : false,
    "keypoololdest" : 1657407652,
    "keypoolsize" : 2,
    "paytxfee" : 0.00000000,
    "relayfee" : 0.00000000,
    "errors" : ""
}
chain1:
chain1:
chain1:
chain1:
chain1:
chain1: getblockhash161
{"method":"getblockhash161","params":[],"id":1,"chain_name":"chain1"}

error code: -32601
error message:
Method not found
chain1: getblockhash 161
{"method":"getblockhash","params":[161],"id":1,"chain_name":"chain1"}

0094e3255b623c0ecb12bbedefce41be38e3ef86b8a3b941006afcd62e39511f
chain1:
chain1:
chain1:
chain1:
chain1:
```

multichain-1.0.1 : multichain-cli

multichain-1.0.1 : multichain-cli — Konsole

File Edit View Bookmarks Settings Help

```
}
chain1:
chain1:
chain1:
chain1:
chain1:
chain1: getblockhash161
{"method":"getblockhash161","params":[],"id":1,"chain_name":"chain1"}

error code: -32601
error message:
Method not found
chain1: getblockhash 161
{"method":"getblockhash","params":[161],"id":1,"chain_name":"chain1"}

0094e3255b623c0ecb12bbedefce41be38e3ef86b8a3b941006afcd62e39511f
chain1:
chain1:
chain1:
chain1:
chain1:
chain1: getblock 0094e3255b623c0ecb12bbedefce41be38e3ef86b8a3b941006afcd62e39511f
{"method":"getblock","params":["0094e3255b623c0ecb12bbedefce41be38e3ef86b8a3b941006afcd62e39511f"],"id":1,"ch
ain_name":"chain1"}

{
    "hash" : "0094e3255b623c0ecb12bbedefce41be38e3ef86b8a3b941006afcd62e39511f",
    "miner" : "18k2gqmqeDPJi2JWKsxazbjTwUdJinwRcH8Dxf",
    "confirmations" : 3,
    "size" : 267,
    "height" : 161,
    "version" : 3,
    "merkleroot" : "3e665b522c8ff37f40473902cf70ecfccc0dbedf9f78bb24eb8e99f8c3876011",
    "tx" : [
        "3e665b522c8ff37f40473902cf70ecfccc0dbedf9f78bb24eb8e99f8c3876011"
    ],
    "time" : 1657410598,
    "nonce" : 26,
    "bits" : "2000ffff",
    "difficulty" : 0.00000006,
    "chainwork" : "000000000000000000000000000000000000000000000000000000000000a200",
    "previousblockhash" : "004e005ebf9cbbf53ae580e115219d47a4e2484b2b37214122c22ec02aa8046d",
    "nextblockhash" : "00104c3a77dd22ed914cbf70eceb5a6937348a2fe986bda92ff42e4243cd3e10"
}
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
chain1:
```

multichain-1.0.1 : multichain-cli

Conclusion :- All in all we saw introducing multichain in Linux then we survey the multichain params.dat and comprehend what are the boundaries that we can utilize according to our prerequisites. It lets us know what order we really want to run while we having 27 a blunder or we type wrong order or it guide us what order would it be a good idea for us we want to run that is extraordinary element of multichain. All client authorization in own grasp who made multichain so he can give admittance to other who can join multichain or send or get resources. We can see rundown of authorization of any client by "listpermissions" order. We can produce quite a few locations to getresource so nobody can follow back. Then we perceive how we can send resource with information (information could be whatever depicts for what reason resource has been sent). Then, at that point, at last we how to send nuclear trade in that we can send exchange with secure way it mean on the off chance that shipper send exchange, beneficiary unquestionable requirement get exchange this is finished by some of order which are ex: preparelockunspent '{"asset2":1}' (asset2 is given of resource, 1 is send resource in one single part), createrawexchange (it will make a hex code that can be unravel by decoderawexchange), appendrawexchange hex code(which is we interpret and got another later decoderawexchange order.) address (recipient preparelockunspent address) 1 '{"asset2":1}'. We additionally perceive how we can get block exchange data in light of blockhash (blockhash will be found in view of block number).