

Lending Smart Contract 2-of-3 Multi-Signature Facility

by Douglas Bebbler - 01 December 2016

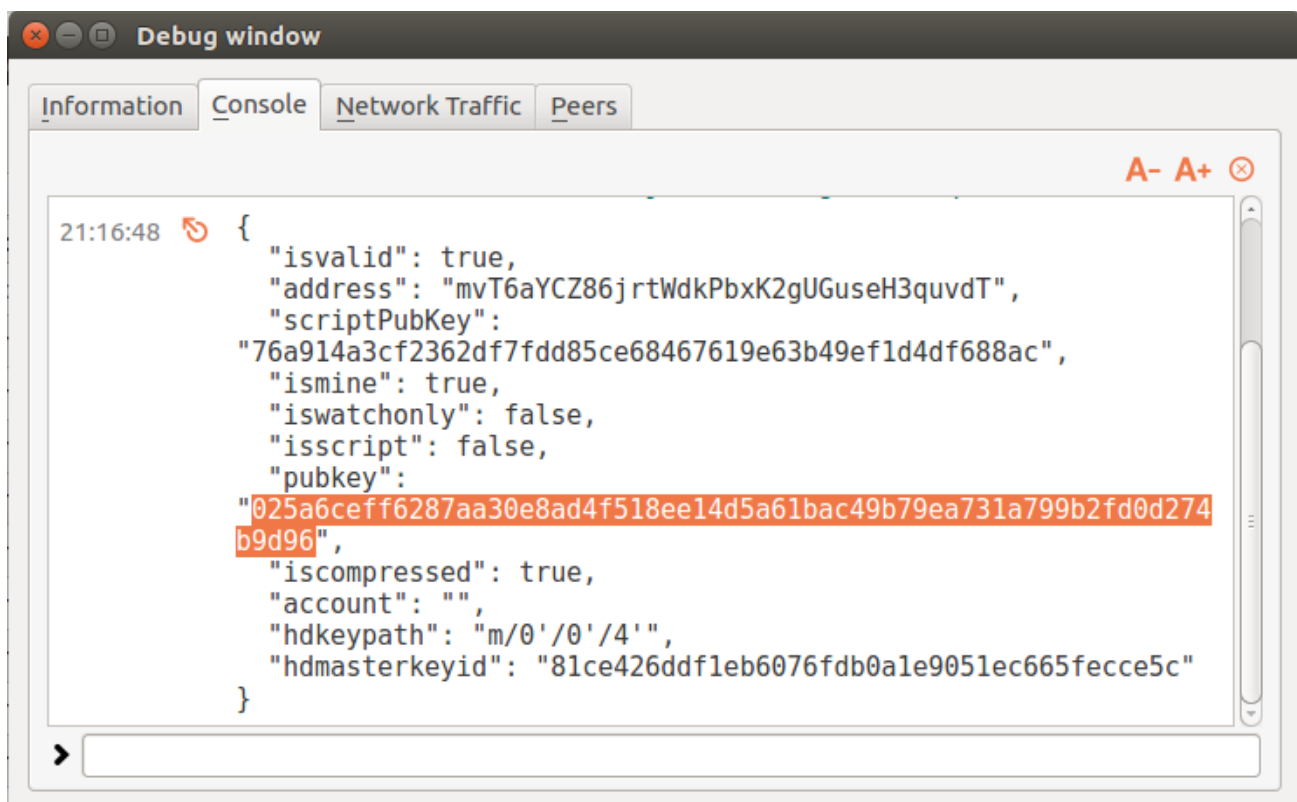
(Many of the steps outlined here will be automated in the background in the software therefore users will not be burdened with a complex user experience. The steps are illustrated here with the bitcoin-qt software simply to illustrate the necessary steps in a comprehensive manner so that readers can follow along.)

All parties to the smart contract need to provide a bitcoin public key for the 2-of-3 multi-signature facility. To obtain a public key for this purpose, each party should generate a new bitcoin address in their wallet. Then issue the **validateaddress** command to obtain their public key.

For example, we will demonstrate this process for the borrower. The borrower creates a new bitcoin address: `mvT6aYCZ86jrtWdkPbxK2gUGuseH3quvdT`

Now the borrower goes to the **Debug window** in the bitcoin-qt application and issues the `validateaddress` command as shown below:





Borrower pub key: 025a6ceff6287aa30e8ad4f518ee14d5a61bac49b79ea731a799b2fd0d274b9d96

Lender pub key: 021ce599ec4b3e18e3dfd6a7ae157464cd920f96eb7a7ea24a99e15d2c1f39d85

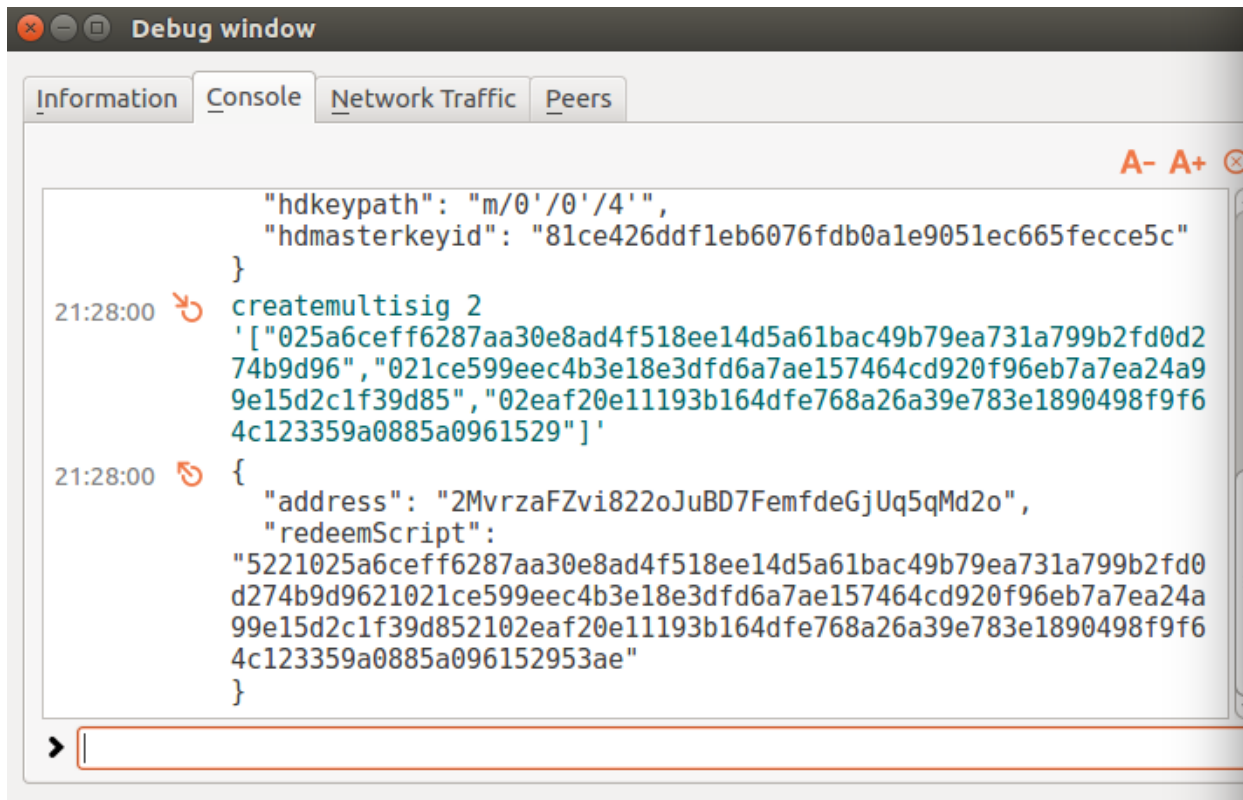
Arbiter pub key: 02eaf20e11193b164dfe768a26a39e783e1890498f9f64c123359a0885a0961529

The borrower and lender need to provide their public keys to the Arbiter software agent to create the lending smart contract.

The Arbiter software agent will then create the 2-of-3 multi-signature bitcoin address to secure the borrowers collateral deposit. The Arbiter software agent does this using the **createmultisig** command. The parameters to the createmultisig command are the number of key signatures necessary to spend the multi-sig funds (2), and the 3 public keys designated as the set of which two are necessary to unlock the funds i.e., 2-of-3. Here is the command:

```
createmultisig 2
'["025a6ceff6287aa30e8ad4f518ee14d5a61bac49b79ea731a799b2fd0d274b9d96","021ce599e
ec4b3e18e3dfd6a7ae157464cd920f96eb7a7ea24a99e15d2c1f39d85","02eaf20e11193b164dfe7
68a26a39e783e1890498f9f64c123359a0885a0961529"]'
```

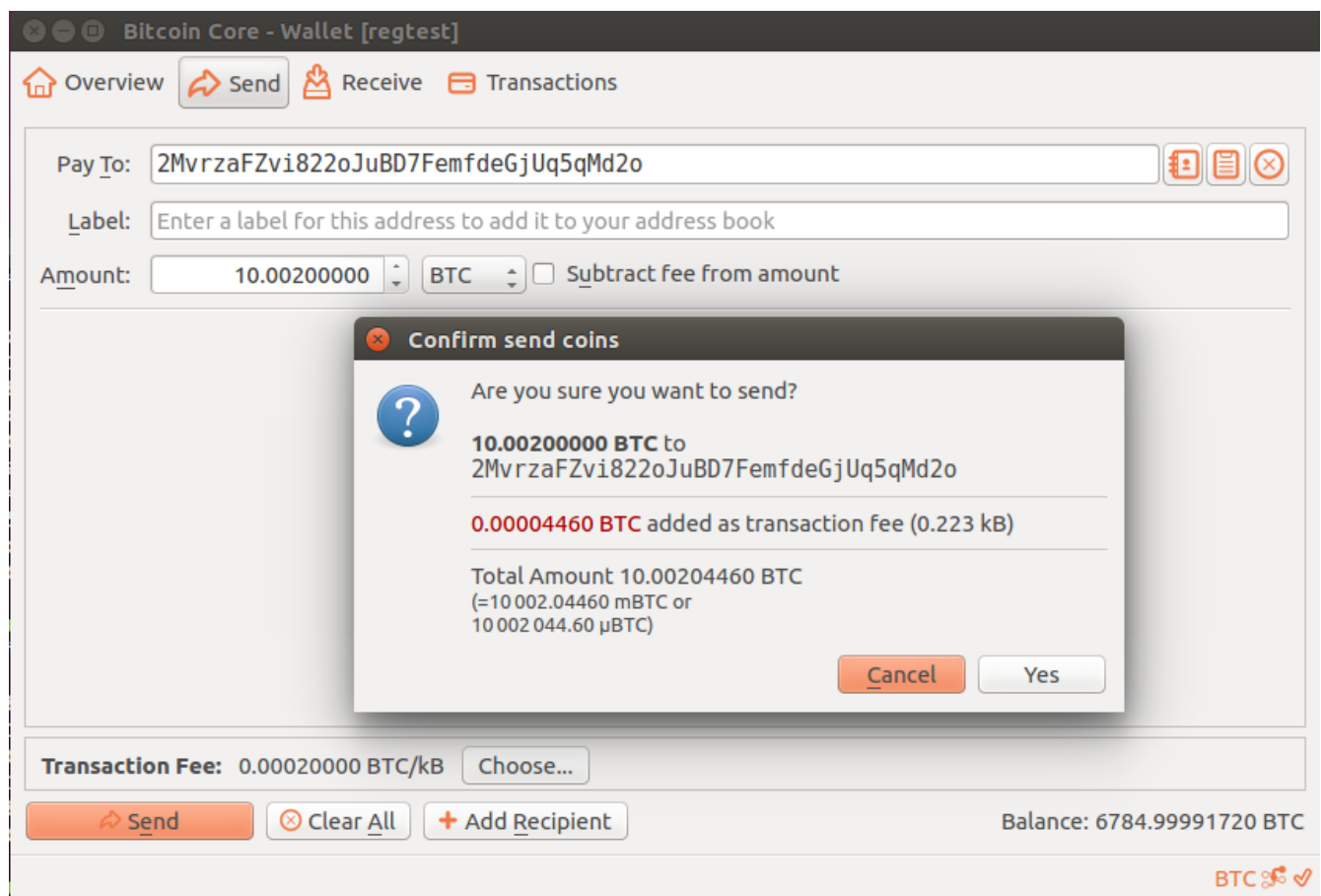
Here is the output from the bitcoin-qt Debug window:



Now the borrower can send his collateral to the multi-signature address:

`2MvrzaFZvi822oJuBD7FemfdeGjUq5qMd2o`

For example, lets have the borrower deposit 10.002 BTC as collateral into the lending smart contract multi-signature address:



Once sent here is a screenshot of the transaction:

Bitcoin Core - Wallet [regtest]

Overview Send Receive Transactions

All All Enter address or label to search Min amount

Date	Type	Label	Amount (BTC)
11/30/16 21:34	Sent to	(2MvrzaFZvi822oJuBD7FemfdeGjUq5qMd2o)	-10.00204460
11/30/16 20:50			0.00000000
11/30/16 20:50			0.00000000
11/30/16 20:50			0.00000000
11/30/16 20:50			0.00000000
11/30/16 20:50			0.00000000
11/30/16 20:50			0.00000000
11/30/16 20:50			0.00000000
11/30/16 17:40			0.00000000
11/30/16 17:40			0.00000000
11/30/16 17:46	Mined	(mjw1AhTz6M5uCzPemk5Be5V14Yc9PjS7pN)	[25.00000000]
11/30/16 17:46	Mined	(mjw1AhTz6M5uCzPemk5Be5V14Yc9PjS7pN)	[25.00000000]
11/30/16 17:46	Mined	(mjw1AhTz6M5uCzPemk5Be5V14Yc9PjS7pN)	[25.00000000]

Export

BTC

Details for 472566f21950944fb432a3a2bb6e15b730b352eea7b14044af94879e520208ed

Status: 0/unconfirmed, in memory pool, broadcast through 2 node(s)

Date: 11/30/16 21:34

To: 2MvrzaFZvi822oJuBD7FemfdeGjUq5qMd2o

Debit: -10.00200000 BTC

Transaction fee: -0.00004460 BTC

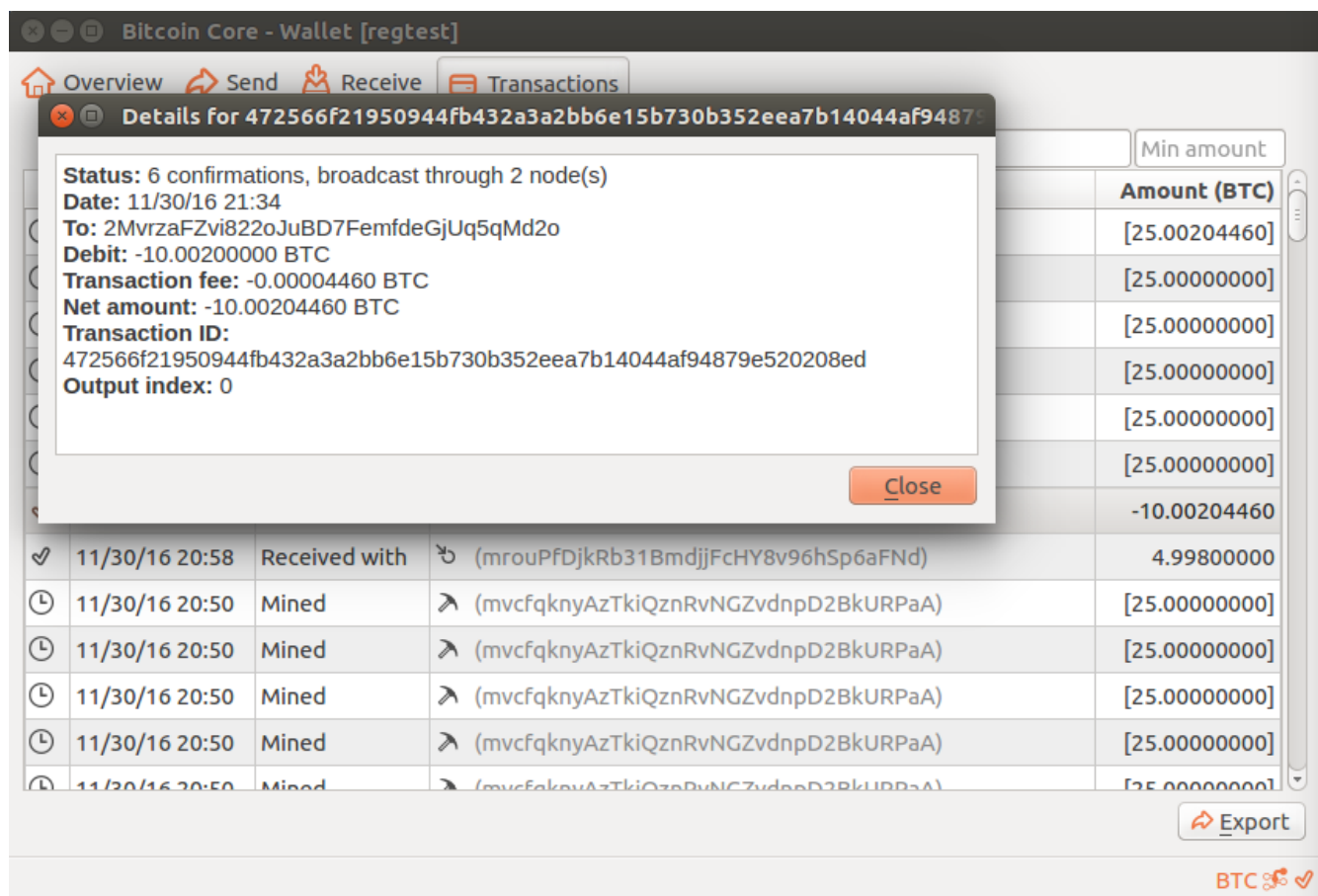
Net amount: -10.00204460 BTC

Transaction ID: 472566f21950944fb432a3a2bb6e15b730b352eea7b14044af94879e520208ed

Output index: 0

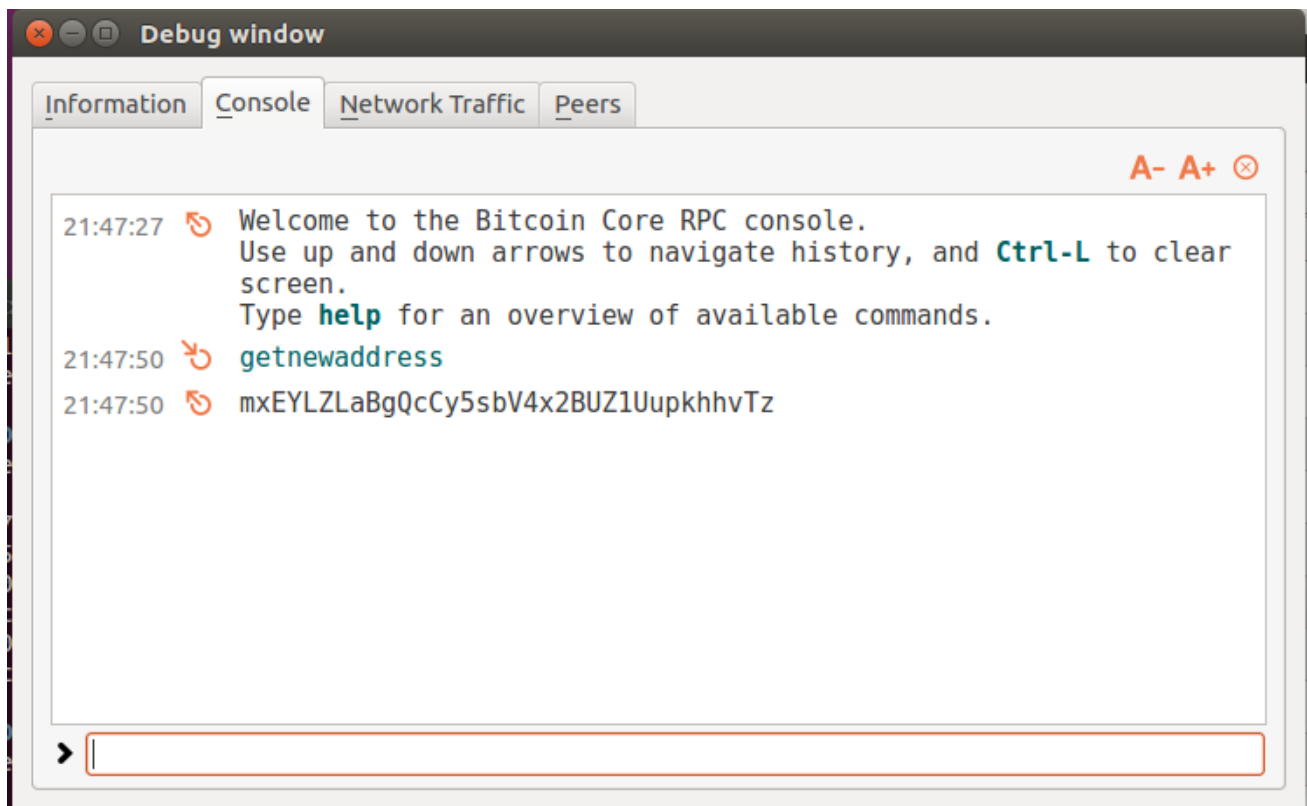
Close

Once this transaction has confirmations all parties can be assured that the collateral is safely secured for purposes of the smart contract (see figure below).



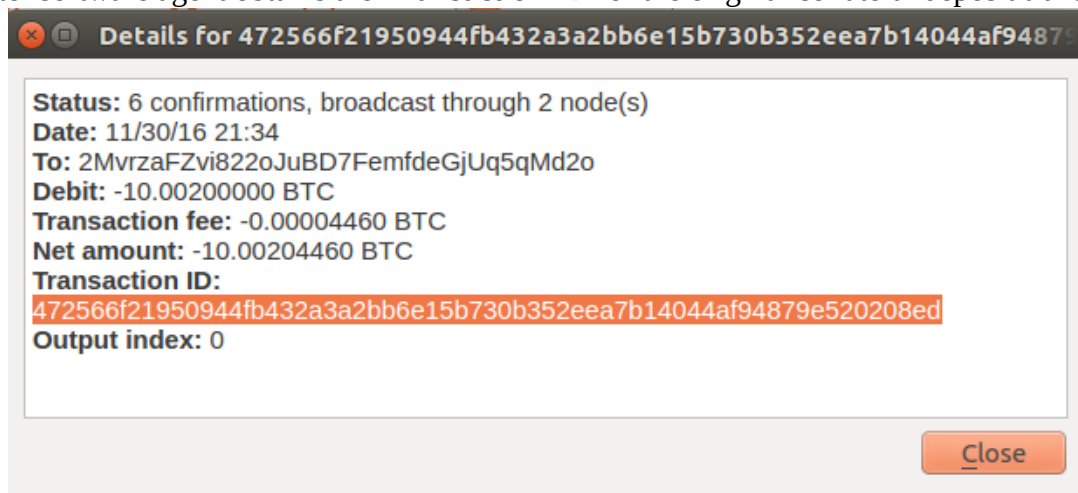
Now when the loan is complete, the collateral needs to be returned to a provided bitcoin address. If the borrower defaults, the collateral will be sent to an address of the lender. If the borrower does not default and executes the contract according to the terms, the collateral will be sent to a bitcoin address of the borrower. Basically what follows is a single set of coded commands the only variables are the address to where the collateral is sent and the two keys used to sign the collateral flow transaction.

For example purposes, let's say everything went according to plan and the borrower is getting his collateral back. In this case the borrower provides a new bitcoin address to receive his collateral. The borrower creates a new address to receive his collateral:



Collateral return address: mxEYLZLaBgQcCy5sbV4x2BUZ1UupkhhvTz

The Arbiter software agent obtains the **Transaction ID** for the original collateral deposit transaction:



The Arbiter software agent then inspects the raw transaction for that Transaction ID using the command:

```
getrawtransaction  
472566f21950944fb432a3a2bb6e15b730b352eea7b14044af94879e520208ed 1
```

The output of the command from the bitcoin-qt Debug window below:

21:57:51

[OBJ]

getrawtransaction

472566f21950944fb432a3a2bb6e15b730b352eea7b14044af94879e520208ed 1

21:57:51

[OBJ]

```
{
  "hex":
    "01000000012ce2ad70273e0b71f84b366e488b60e5e68ac6f1f5e96356392a207107460aae0000
    00006a47304402202b489936a4b65e162c9436fb7c08ec59455dacb414474d9ba88bcf3817fc871
    502204fc5897a6c20f10d001928a404309b265d44a3889756abef01c03b25527d838b01210268d
    b342e58791865b46e6d56d297829d10363a17b37203e5c67d3272d49e67b8feffffff0240d79d3b0
    000000017a91427ac277c9a87be168956edeaf7f9c5609b6a5b9787fcb9ff94000000001976a9140
    d395c69e138b015dee0d833cd83a06f6c1b82ae88acec000000",
  "txid": "472566f21950944fb432a3a2bb6e15b730b352eea7b14044af94879e520208ed",
  "hash": "472566f21950944fb432a3a2bb6e15b730b352eea7b14044af94879e520208ed",
  "size": 223,
  "vsize": 223,
  "version": 1,
  "locktime": 236,
  "vin": [
    {
      "txid": "ae0a460771202a395663e9f5f1c68ae6e5608b486e364bf8710b3e2770ade22c",
      "vout": 0,
      "scriptSig": {
        "asm":
          "304402202b489936a4b65e162c9436fb7c08ec59455dacb414474d9ba88bcf3817fc871502204f
          c5897a6c20f10d001928a404309b265d44a3889756abef01c03b25527d838b[ALL]
          0268db342e58791865b46e6d56d297829d10363a17b37203e5c67d3272d49e67b8",
        "hex":
          "47304402202b489936a4b65e162c9436fb7c08ec59455dacb414474d9ba88bcf3817fc87150220
          4fc5897a6c20f10d001928a404309b265d44a3889756abef01c03b25527d838b01210268db342e
          58791865b46e6d56d297829d10363a17b37203e5c67d3272d49e67b8"
      },
      "sequence": 4294967294
    }
  ],
  "vout": [
    {
      "value": 10.00200000,
      "n": 0,
      "scriptPubKey": {
        "asm": "OP_HASH160 27ac277c9a87be168956edeaf7f9c5609b6a5b97 OP_EQUAL",
        "hex": "a91427ac277c9a87be168956edeaf7f9c5609b6a5b9787",
        "reqSigs": 1,
        "type": "scripthash",
        "addresses": [
```



```

    "2MvrzaFZvi822oJuBD7FemfdeGjUq5qMd2o"
  ]
}
},
{
  "value": 24.99787260,
  "n": 1,
  "scriptPubKey": {
    "asm": "OP_DUP OP_HASH160 0d395c69e138b015dee0d833cd83a06f6c1b82ae
OP_EQUALVERIFY OP_CHECKSIG",
    "hex": "76a9140d395c69e138b015dee0d833cd83a06f6c1b82ae88ac",
    "reqSigs": 1,
    "type": "pubkeyhash",
    "addresses": [
      "mgiso4PnWZKutCBk5BSXTMKqCAMdAP14rX"
    ]
  }
}
],
"blockhash": "47e0b5634013daa8176c33e4e6c53d693d32ce46b3b21b5bf330c4071f80c3b6",
"confirmations": 6,
"time": 1480563482,
"blocktime": 1480563482
}

```

The key information necessary for the next command is highlighted in yellow above. Basically, this means we need to direct the appropriate output of that transaction as an input for the multi-sig spend transaction.

The Arbiter software agent then issues a **createrawtransaction** command to create the transaction to return the collateral to the borrower:

```

createrawtransaction
'[{ "txid": "472566f21950944fb432a3a2bb6e15b730b352eea7b14044af94879e520208ed", "vout": 0 }]' {"mxEYLZLaBgQcCy5sbV4x2BUZ1UupkhvTz": 10.0}

```

From the bitcoin-qt Debug window:


```

221025a6ceff6287aa30e8ad4f518ee14d5a61bac49b79ea731a799b2fd0d274b9d9621021ce599
eec4b3e18e3dfd6a7ae157464cd920f96eb7a7ea24a99e15d2c1f39d852102eaf20e11193b164dfe7
68a26a39e783e1890498f9f64c123359a0885a096152953aeffffffff0100ca9a3b000000001976a91
4b75f7c1b0f100bab19d017b42a9dee75b628104388ac00000000",
  "complete": false,
  "errors": [
    {
      "txid": "472566f21950944fb432a3a2bb6e15b730b352eea7b14044af94879e520208ed",
      "vout": 0,
      "scriptSig":
"004730440220448e0b882f5d388ac721b6b54e10a8280a979b2358ac2a620493f9a4d252a88f0
220115814a559581e7668d222facca4c839aa5005ec487a27376ca6d9442bd29f0c014c69522102
5a6ceff6287aa30e8ad4f518ee14d5a61bac49b79ea731a799b2fd0d274b9d9621021ce599eec4b3
e18e3dfd6a7ae157464cd920f96eb7a7ea24a99e15d2c1f39d852102eaf20e11193b164dfe768a26
a39e783e1890498f9f64c123359a0885a096152953ae",
      "sequence": 4294967295,
      "error": "Operation not valid with the current stack size"
    }
  ]
}

```

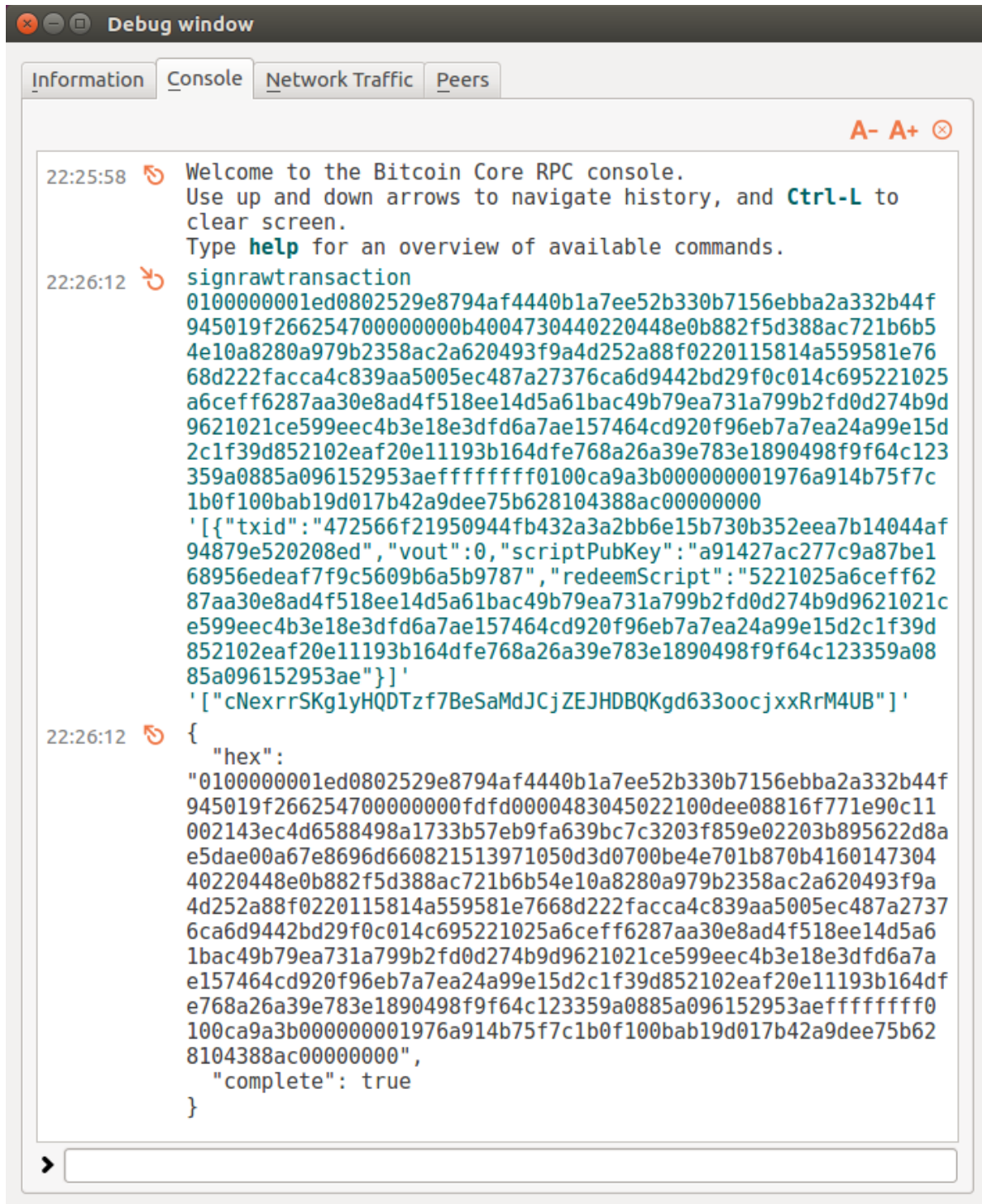
Now, the borrower needs to sign the partially signed raw transaction (highlighted in yellow above):

```

signrawtransaction
0100000001ed0802529e8794af4440b1a7ee52b330b7156ebba2a332b44f945019f26625470000
0000b4004730440220448e0b882f5d388ac721b6b54e10a8280a979b2358ac2a620493f9a4d252
a88f0220115814a559581e7668d222facca4c839aa5005ec487a27376ca6d9442bd29f0c014c695
221025a6ceff6287aa30e8ad4f518ee14d5a61bac49b79ea731a799b2fd0d274b9d9621021ce599
eec4b3e18e3dfd6a7ae157464cd920f96eb7a7ea24a99e15d2c1f39d852102eaf20e11193b164dfe7
68a26a39e783e1890498f9f64c123359a0885a096152953aeffffffff0100ca9a3b000000001976a91
4b75f7c1b0f100bab19d017b42a9dee75b628104388ac00000000
'[{ "txid": "472566f21950944fb432a3a2bb6e15b730b352eea7b14044af94879e520208ed", "vout"
:0, "scriptPubKey": "a91427ac277c9a87be168956edeaf7f9c5609b6a5b9787", "redeemScript": "5
221025a6ceff6287aa30e8ad4f518ee14d5a61bac49b79ea731a799b2fd0d274b9d9621021ce599
eec4b3e18e3dfd6a7ae157464cd920f96eb7a7ea24a99e15d2c1f39d852102eaf20e11193b164dfe7
68a26a39e783e1890498f9f64c123359a0885a096152953ae"}]'
'["cNexrrSKg1yHQDTzf7BeSaMdJCjZEHDBQKgd633oocjxxRrM4UB"]'

```

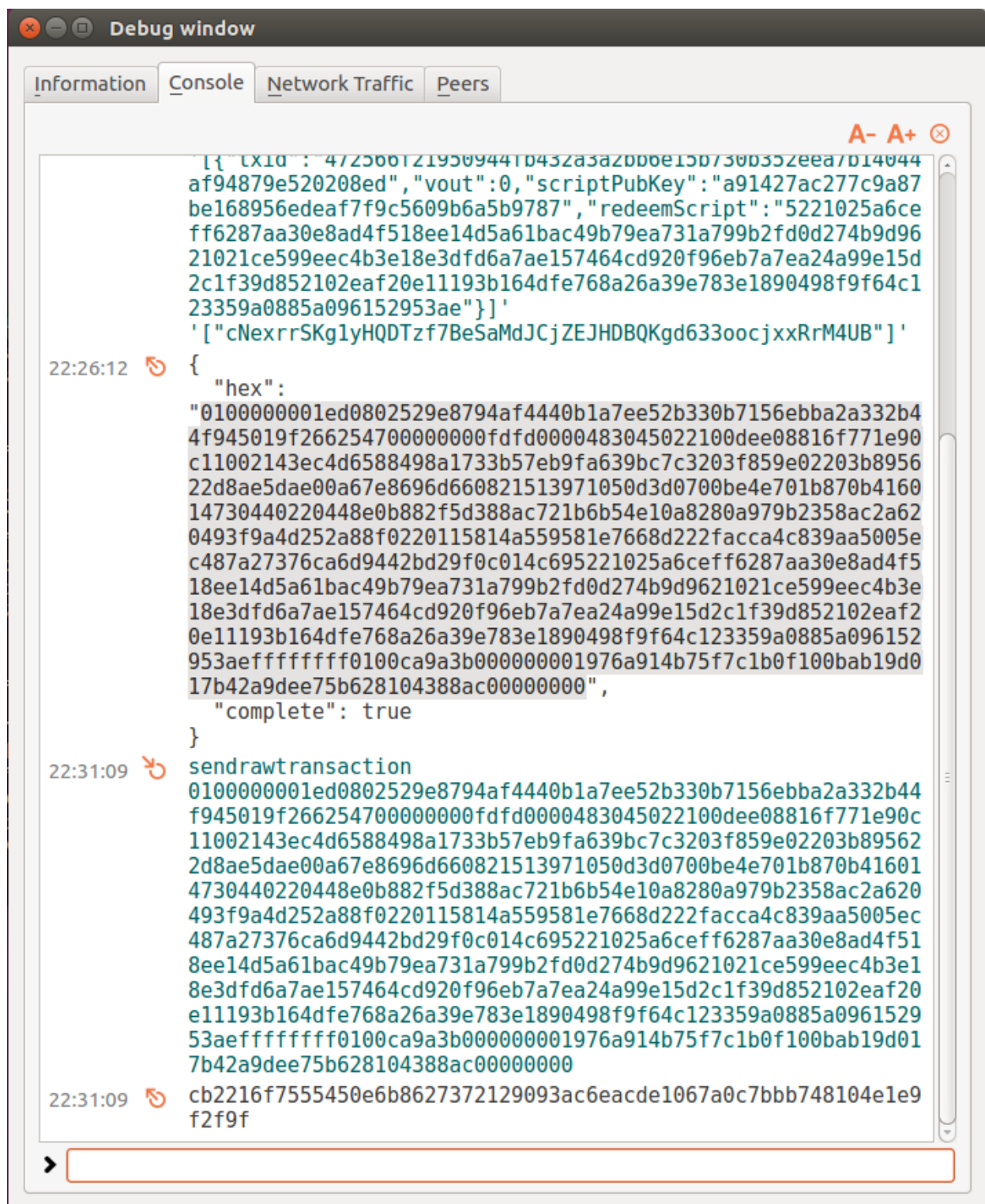
bitcoin-qt Debug window output:



The screenshot shows the Bitcoin-Qt Debug window with the 'Console' tab selected. The output shows a successful transaction signing process. The first message is a welcome message to the Bitcoin Core RPC console. The second message shows the command 'signrawtransaction' being executed, followed by a large hexadecimal string representing the raw transaction. The third message shows the JSON response from the RPC, which includes the signed transaction hex and a 'complete' status of true.

```
22:25:58 Welcome to the Bitcoin Core RPC console.  
Use up and down arrows to navigate history, and Ctrl-L to  
clear screen.  
Type help for an overview of available commands.  
22:26:12 signrawtransaction  
0100000001ed0802529e8794af4440b1a7ee52b330b7156ebba2a332b44f  
945019f2662547000000000b4004730440220448e0b882f5d388ac721b6b5  
4e10a8280a979b2358ac2a620493f9a4d252a88f0220115814a559581e76  
68d222facca4c839aa5005ec487a27376ca6d9442bd29f0c014c695221025  
a6ceff6287aa30e8ad4f518ee14d5a61bac49b79ea731a799b2fd0d274b9d  
9621021ce599eec4b3e18e3dfd6a7ae157464cd920f96eb7a7ea24a99e15d  
2c1f39d852102eaf20e11193b164dfe768a26a39e783e1890498f9f64c123  
359a0885a096152953aefffffffff0100ca9a3b000000001976a914b75f7c  
1b0f100bab19d017b42a9dee75b628104388ac00000000  
'[{"txid": "472566f21950944fb432a3a2bb6e15b730b352eea7b14044af  
94879e520208ed", "vout": 0, "scriptPubKey": "a91427ac277c9a87be1  
68956edeaf7f9c5609b6a5b9787", "redeemScript": "5221025a6ceff62  
87aa30e8ad4f518ee14d5a61bac49b79ea731a799b2fd0d274b9d9621021c  
e599eec4b3e18e3dfd6a7ae157464cd920f96eb7a7ea24a99e15d2c1f39d  
852102eaf20e11193b164dfe768a26a39e783e1890498f9f64c123359a08  
85a096152953ae"}]'  
22:26:12 {  
  "hex":  
    "0100000001ed0802529e8794af4440b1a7ee52b330b7156ebba2a332b44f  
945019f2662547000000000fd0000483045022100dee08816f771e90c11  
002143ec4d6588498a1733b57eb9fa639bc7c3203f859e02203b895622d8a  
e5dae00a67e8696d660821513971050d3d0700be4e701b870b4160147304  
40220448e0b882f5d388ac721b6b54e10a8280a979b2358ac2a620493f9a  
4d252a88f0220115814a559581e7668d222facca4c839aa5005ec487a2737  
6ca6d9442bd29f0c014c695221025a6ceff6287aa30e8ad4f518ee14d5a6  
1bac49b79ea731a799b2fd0d274b9d9621021ce599eec4b3e18e3dfd6a7a  
e157464cd920f96eb7a7ea24a99e15d2c1f39d852102eaf20e11193b164df  
e768a26a39e783e1890498f9f64c123359a0885a096152953aefffffffff0  
100ca9a3b0000000001976a914b75f7c1b0f100bab19d017b42a9dee75b62  
8104388ac00000000",  
    "complete": true  
  }  
}
```

The last thing to do is to send (or broadcast) the signed transaction out to the bitcoin network using the **sendrawtransaction** command:



We see the borrower has received his collateral and the 2-of-3 multi-signature facility of the lending smart contract is now completed.

The screenshot shows the Bitcoin Core - Wallet [regtest] interface. The top navigation bar includes 'Overview', 'Send', 'Receive', and 'Transactions'. Below this, there are filters for 'All' and 'All', a search bar 'Enter address or label to search', and a 'Min amount' field. The main table displays a list of transactions with columns: Date, Type, Label, and Amount (BTC). A transaction with ID 'cb2216f7555450e6b8627372129093ac6eacde1067a0c7bbb74810' is highlighted, and its details are shown in a pop-up window.

Date	Type	Label	Amount (BTC)
11/30/16 22:32	Mined	(mgRYbZ5uKGox9ZFY4A12cp3RkQY8NpeS5E)	[25.00200000]
11/30/16 22:32	Mined	(mgRYbZ5uKGox9ZFY4A12cp3RkQY8NpeS5E)	[25.00000000]
11/30/16 22:32	Mined	(mgRYbZ5uKGox9ZFY4A12cp3RkQY8NpeS5E)	[25.00000000]
11/30/16 22:32	Mined	(mgRYbZ5uKGox9ZFY4A12cp3RkQY8NpeS5E)	[25.00000000]
11/30/16 22:32	Mined	(mgRYbZ5uKGox9ZFY4A12cp3RkQY8NpeS5E)	[25.00000000]
11/30/16 22:32	Mined	(mgRYbZ5uKGox9ZFY4A12cp3RkQY8NpeS5E)	[25.00000000]
11/30/16 22:31	Received with	(mxEYLZLaBgQcCy5sbV4x2BUZ1UupkhvTz)	10.00000000

Details for cb2216f7555450e6b8627372129093ac6eacde1067a0c7bbb74810

Status: 6 confirmations
Date: 11/30/16 22:31
From: unknown
To: mxEYLZLaBgQcCy5sbV4x2BUZ1UupkhvTz (own address)
Credit: 10.00000000 BTC
Net amount: +10.00000000 BTC
Transaction ID: cb2216f7555450e6b8627372129093ac6eacde1067a0c7bbb748104e1e9f2f9f
Output index: 0

High-level Summary:

- All parties supply a public key for the 2-of-3 multi-signature facility
- The Arbiter uses those public keys to create a multi-signature address that collateral can be deposited to
- The borrower deposits collateral into the multi-signature addresses
- Collateral is returned to the borrower or released to the lender based on whether the contract was successfully executed or breached. This is governed by the Arbiter software agent via the collateral flow transaction authorized through the signing of the collateral flow transaction by the corresponding private keys of the initially provided public keys.
- The successfully signed collateral flow transaction is then broadcasted to the Bitcoin network to release the collateral from the multi-signature facility