# Manual procedure to spend time-locked transaction using libbitcoin bx commands.

Miłosz Muszyński 8.3.2021

Based on:

<https://github.com/libbitcoin/libbitcoin-explorer/wiki/How-to-Spend-From-a-Multisig-Address>

Assume we have 1000000 Satoshis balance at 2MwGGufthfjcGKA8KB4vSXoAHHVBJsezJy8, and have private key to it.

bx fetch-balance 2MwGGufthfjcGKA8KB4vSXoAHHVBJsezJy8

## Determine the Funding Transaction and Vout (output index), i.e. – Funding UTXO

bx fetch-tx 2a0990b736e79e1d65ce3e9e25427e36855235829d58c1f2a9eac18142c926a6  
  
transaction  
{  
 hash 2a0990b736e79e1d65ce3e9e25427e36855235829d58c1f2a9eac18142c926a6  
 inputs  
 {  
 input  
 {  
 address\_hash 43da33b4013c107ec381f126afbfb172e0e25b9b  
 previous\_output  
 {  
 hash 5a6c2627f80c4aad2f383c2245aadd5bf267313648691b063a95bf91becc4c0b  
 index 1  
 }  
 script "[3045022100e56d3295de26496c476e926037166bc1094711c593e87490c8b9ed8ed7809892022039f39edee7fc893a8e0500585da9926ccca94cf8b598dfa2bd080d1133c1d8e001] [0314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1d]"  
 sequence 4294967294  
 }  
 }  
 lock\_time 0  
 outputs  
 {  
 output  
 {  
 address\_hash 2c135b63577126ac7164804aa40eb148ce934173  
 script "hash160 [2c135b63577126ac7164804aa40eb148ce934173] equal"  
 value 900000  
 }  
 }  
 version 0  
}

points

{

point

{

hash 2a0990b736e79e1d65ce3e9e25427e36855235829d58c1f2a9eac18142c926a6

index 0

value 900000

}

}

input for the new transaction will be:  
2a0990b736e79e1d65ce3e9e25427e36855235829d58c1f2a9eac18142c926a6:0

## Determine Target Address and Amount to be Sent

the output for the new transaction will be:  
n4eaAFB3GPmrJR4ummYpQmYTx2VaNftuPe 800000

## Encoding new transaction

bx tx-encode -i 2a0990b736e79e1d65ce3e9e25427e36855235829d58c1f2a9eac18142c926a6:0:0 -o n4eaAFB3GPmrJR4ummYpQmYTx2VaNftuPe:800000 -l 1615161540

-i means input

-o means output

-l means lock\_time

Note TXHASH:INDEX:SEQUENCE format for -i.

bx tx-decode 0100000001a626c94281c1eaa9f2c1589d82355285367e42259e3ece651d9ee736b790092a0000000000000000000100350c00000000001976a914fdbbbe6062fef2fca812e404e3dcb43dcdb4108888acc4684560

transaction  
{  
 hash 85ff5d95d58fa7438d9db0305af4fbec0350c14b301a2116e048045611d62173  
 inputs  
 {  
 input  
 {  
 previous\_output  
 {  
 hash 2a0990b736e79e1d65ce3e9e25427e36855235829d58c1f2a9eac18142c926a6  
 index 0  
 }  
 script ""  
 sequence 0  
 }  
 }  
 lock\_time 1615161540  
 outputs  
 {  
 output  
 {  
 address\_hash fdbbbe6062fef2fca812e404e3dcb43dcdb41088  
 script "dup hash160 [fdbbbe6062fef2fca812e404e3dcb43dcdb41088] equalverify checksig"  
 value 800000  
 }  
 }  
 version 1  
}

Notice that transactions.inputs[0].script is empty. This means that the input has not been endorsed.

## Create Endorsement

Create the endorsement for the first input 85ff5d95d58fa...:0 of the new transaction, using the first private key, the multisig script and the new transaction.  
  
bx input-sign 156bc5c6f50244b6745e7280671697e8eb703a81a7daf1180ec65d957bff429e "[c4684560] checklocktimeverify drop [0314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1d] checksig" 0100000001a626c94281c1eaa9f2c1589d82355285367e42259e3ece651d9ee736b790092a0000000000000000000100350c00000000001976a914fdbbbe6062fef2fca812e404e3dcb43dcdb4108888acc4684560

Response:

3044022024048cd26f0d493173c4c1e15be7fc4bb0c9f91bbba422d46d09b910ec28c0ac02202477ce166b20b13aee4e997c5e2ffbdd8d274d06066307690f75dd7dcc5a3a6a01

BX Input-sign creates an endorsement for a transaction input. Endorsement = signature.

Takes:

* private key
* previous output script
* transaction

## Encode Script

Encode the (multisig, CLTV etc) script, for embedding in the endorsement script.  
  
bx script-encode "[c4684560] checklocktimeverify drop [0314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1d] checksig"  
  
04c4684560b175210314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1dac

BX Script-encode encodes plain text script into HEX.

## Assign Endorsement Script to Input

Create an endorsement script using the endorsement and the encoded script and assign it to the first input of the transaction.

Assigns a script to an existing transaction input.  
  
bx input-set " [3044022024048cd26f0d493173c4c1e15be7fc4bb0c9f91bbba422d46d09b910ec28c0ac02202477ce166b20b13aee4e997c5e2ffbdd8d274d06066307690f75dd7dcc5a3a6a01] [04c4684560b175210314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1dac]" 0100000001a626c94281c1eaa9f2c1589d82355285367e42259e3ece651d9ee736b790092a0000000000000000000100350c00000000001976a914fdbbbe6062fef2fca812e404e3dcb43dcdb4108888acc4684560  
  
0100000001a626c94281c1eaa9f2c1589d82355285367e42259e3ece651d9ee736b790092a0000000073473044022024048cd26f0d493173c4c1e15be7fc4bb0c9f91bbba422d46d09b910ec28c0ac02202477ce166b20b13aee4e997c5e2ffbdd8d274d06066307690f75dd7dcc5a3a6a012a04c4684560b175210314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1dac000000000100350c00000000001976a914fdbbbe6062fef2fca812e404e3dcb43dcdb4108888acc4684560  
  
bx tx-decode 0100000001a626c94281c1eaa9f2c1589d82355285367e42259e3ece651d9ee736b790092a0000000073473044022024048cd26f0d493173c4c1e15be7fc4bb0c9f91bbba422d46d09b910ec28c0ac02202477ce166b20b13aee4e997c5e2ffbdd8d274d06066307690f75dd7dcc5a3a6a012a04c4684560b175210314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1dac000000000100350c00000000001976a914fdbbbe6062fef2fca812e404e3dcb43dcdb4108888acc4684560

transaction

{

hash 040ab4ea202cfee98ae29cc5478462f23216585413f1ca841babf4124af68502

inputs

{

input

{

address\_hash 2c135b63577126ac7164804aa40eb148ce934173

previous\_output

{

hash 2a0990b736e79e1d65ce3e9e25427e36855235829d58c1f2a9eac18142c926a6

index 0

}

script "[3044022024048cd26f0d493173c4c1e15be7fc4bb0c9f91bbba422d46d09b910ec28c0ac02202477ce166b20b13aee4e997c5e2ffbdd8d274d06066307690f75dd7dcc5a3a6a01] [04c4684560b175210314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1dac]"

sequence 0

}

}

lock\_time 1615161540

outputs

{

output

{

address\_hash fdbbbe6062fef2fca812e404e3dcb43dcdb41088

script "dup hash160 [fdbbbe6062fef2fca812e404e3dcb43dcdb41088] equalverify checksig"

value 800000

}

}

version 1

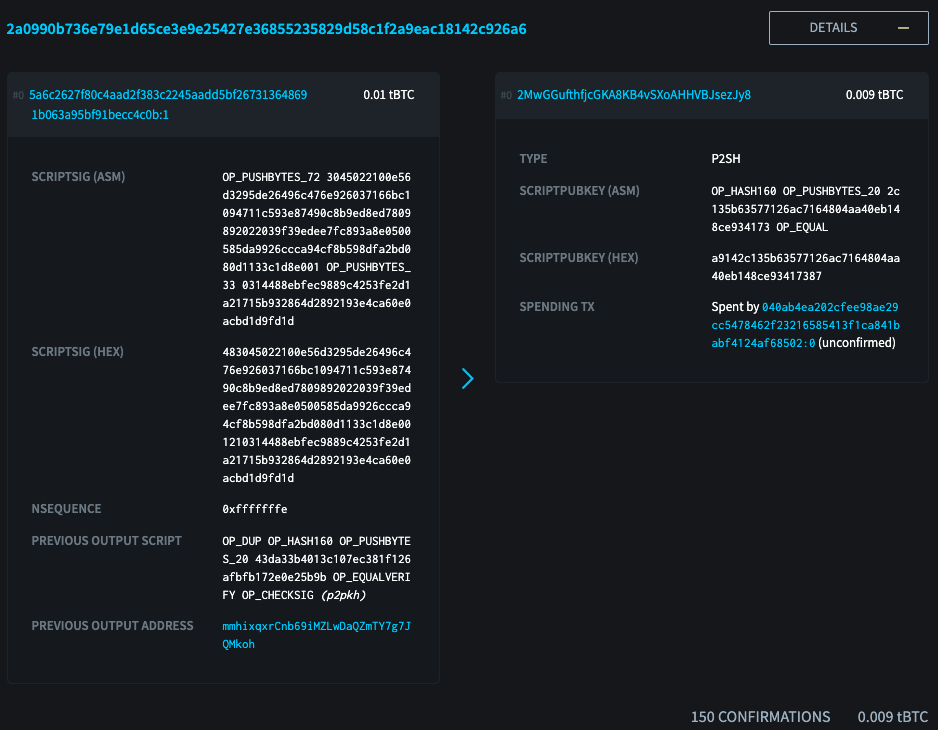
}

Notice that the endorsement script has been applied to transaction.inputs[0].script and that transaction.hash has been updated.  
  
Validate the endorsement of the transaction's first input, using the public key, multisig script, first endorsement and transaction (optional).  
  
bx input-validate 0314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1d "[c4684560] checklocktimeverify drop [0314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1d] checksig" 3044022024048cd26f0d493173c4c1e15be7fc4bb0c9f91bbba422d46d09b910ec28c0ac02202477ce166b20b13aee4e997c5e2ffbdd8d274d06066307690f75dd7dcc5a3a6a01 0100000001a626c94281c1eaa9f2c1589d82355285367e42259e3ece651d9ee736b790092a0000000073473044022024048cd26f0d493173c4c1e15be7fc4bb0c9f91bbba422d46d09b910ec28c0ac02202477ce166b20b13aee4e997c5e2ffbdd8d274d06066307690f75dd7dcc5a3a6a012a04c4684560b175210314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1dac000000000100350c00000000001976a914fdbbbe6062fef2fca812e404e3dcb43dcdb4108888acc4684560

The endorsement is valid

Validate the transaction against the blockchain (optional).  
  
bx validate-tx 0100000001a626c94281c1eaa9f2c1589d82355285367e42259e3ece651d9ee736b790092a0000000073473044022024048cd26f0d493173c4c1e15be7fc4bb0c9f91bbba422d46d09b910ec28c0ac02202477ce166b20b13aee4e997c5e2ffbdd8d274d06066307690f75dd7dcc5a3a6a012a04c4684560b175210314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1dac000000000100350c00000000001976a914fdbbbe6062fef2fca812e404e3dcb43dcdb4108888acc4684560

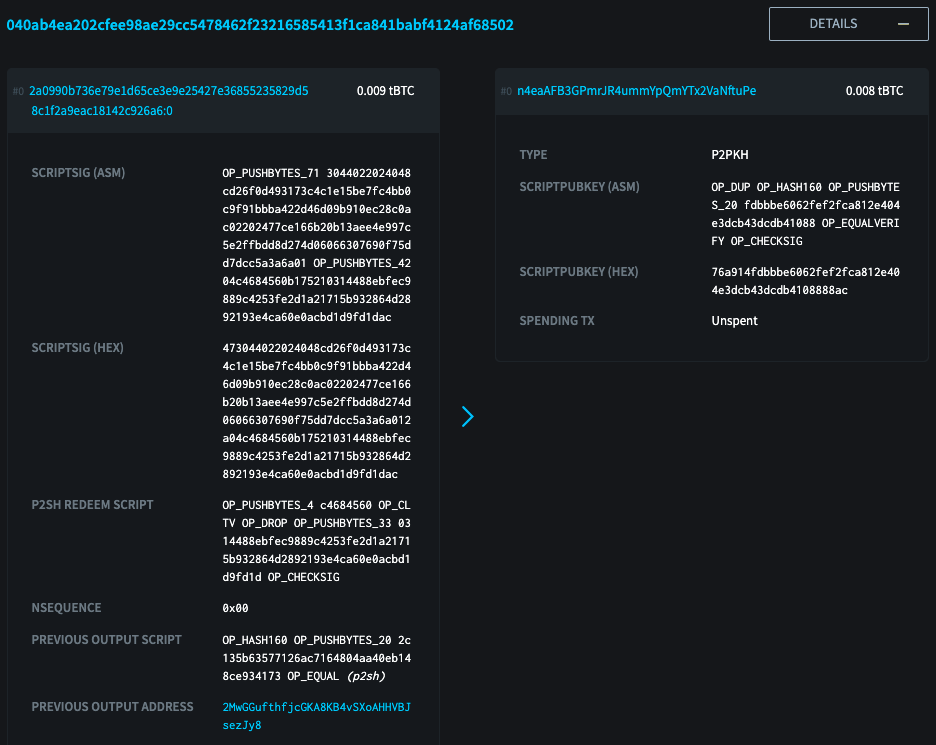
## Funding Transaction – Locking to HODL



We can see this is a P2SH transaction, script hash is a9142c135…

It is spent by the transaction described in the following section.

## HODL Spending Transaction



We can see previous transaction 2a099… was spent by this transaction (040ab4…) using redeem script with OP\_CLTV. Unlocking script is shown as SCRIPTSIG (ASM/HEX). NSEQUENCE is 0 as OP\_CLTV does not approve final inputs. Similarly, transaction lock\_time, not show here, must be set to the original CLTV locktime or later.

As a remainder, the transaction looks as follows:

{ hash 040ab4ea202cfee98ae29cc5478462f23216585413f1ca841babf4124af68502

inputs

{ input

{. address\_hash 2c135b63577126ac7164804aa40eb148ce934173

previous\_output

{

hash 2a0990b736e79e1d65ce3e9e25427e36855235829d58c1f2a9eac18142c926a6

index 0

}

script "[3044022024048cd26f0d493173c4c1e15be7fc4bb0c9f91bbba422d46d09b910ec28c0ac02202477ce166b20b13aee4e997c5e2ffbdd8d274d06066307690f75dd7dcc5a3a6a01] [04c4684560b175210314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1dac]"

sequence 0

}

}

lock\_time 1615161540

outputs

{. output

{. address\_hash fdbbbe6062fef2fca812e404e3dcb43dcdb41088

script "dup hash160 [fdbbbe6062fef2fca812e404e3dcb43dcdb41088] equalverify checksig"

value 800000

}

}

version 1}

## Remarks

If we know the redeem script, which is like:

"[c4684560] checklocktimeverify drop [0314488ebfec9889c4253fe2d1a21715b932864d2892193e4ca60e0acbd1d9fd1d] checksig

We only need to know public hex key to fill out the script, but the biggest caveat is the locktime. If we forget locktime, there is no way to recreate redeem script from the hash. So in order to be able to spent our locked money we need:

* private key to the funding address (where funding tx comes from), this implies having public key
* locktime

That’s it. TBD: verify and triple check if this is true.