Bitcoin Explorer (bx) Commands

2022-03-04

# Bitcoin Explorer (bx) Commands

Bitcoin Explorer (bx) is a command-line tool that offers a variety of commands for key management and transaction construction. It is part of the libbitcoin bitcoin library.

Usage: bx COMMAND [--help]  
  
Info: The bx commands are:  
  
address-decode  
address-embed  
address-encode  
address-validate  
base16-decode  
base16-encode  
base58-decode  
base58-encode  
base58check-decode  
base58check-encode  
base64-decode  
base64-encode  
bitcoin160  
bitcoin256  
btc-to-satoshi  
ec-add  
ec-add-secrets  
ec-multiply  
ec-multiply-secrets  
ec-new  
ec-to-address  
ec-to-public  
ec-to-wif  
fetch-balance  
fetch-header  
fetch-height  
fetch-history  
fetch-stealth  
fetch-tx  
fetch-tx-index  
hd-new  
hd-private  
hd-public  
hd-to-address  
hd-to-ec  
hd-to-public  
hd-to-wif  
help  
input-set  
input-sign  
input-validate  
message-sign  
message-validate  
mnemonic-decode  
mnemonic-encode  
ripemd160  
satoshi-to-btc  
script-decode  
script-encode  
script-to-address  
seed  
send-tx  
send-tx-node  
send-tx-p2p  
settings  
sha160  
sha256  
sha512  
stealth-decode  
stealth-encode  
stealth-public  
stealth-secret  
stealth-shared  
tx-decode  
tx-encode  
uri-decode  
uri-encode  
validate-tx  
watch-address  
wif-to-ec  
wif-to-public  
wrap-decode  
wrap-encode

For more information, see the [Bitcoin Explorer homepage](https://github.com/libbitcoin/libbitcoin-explorer) and [Bitcoin Explorer user documentation](https://github.com/libbitcoin/libbitcoin-explorer/wiki).

# Examples of bx Command Use

Let’s look at some examples of using Bitcoin Explorer commands to experiment with keys and addresses.

Generate a random "seed" value using the seed command, which uses the operating system’s random number generator. Pass the seed to the ec-new command to generate a new private key. We save the standard output into the file *private\_key*:

$ bx seed | bx ec-new > private\_key  
$ cat private\_key  
73096ed11ab9f1db6135857958ece7d73ea7c30862145bcc4bbc7649075de474

Now, generate the public key from that private key using the ec-to-public command. We pass the *private\_key* file into the standard input and save the standard output of the command into a new file *public\_key*:

$ bx ec-to-public < private\_key > public\_key  
$ cat public\_key  
02fca46a6006a62dfdd2dbb2149359d0d97a04f430f12a7626dd409256c12be500

We can reformat the public\_key as an address using the ec-to-address command. We pass the *public\_key* into standard input:

$ bx ec-to-address < public\_key  
17re1S4Q8ZHyCP8Kw7xQad1Lr6XUzWUnkG

Keys generated in this manner produce a type-0 nondeterministic wallet. That means that each key is generated from an independent seed. Bitcoin Explorer commands can also generate keys deterministically, in accordance with BIP-32. In this case, a "master" key is created from a seed and then extended deterministically to produce a tree of subkeys, resulting in a type-2 deterministic wallet.

First, we use the seed and hd-new commands to generate a master key that will be used as the basis to derive a hierarchy of keys:

$ bx seed > seed  
$ cat seed  
eb68ee9f3df6bd4441a9feadec179ff1  
  
$ bx hd-new < seed > master  
$ cat master  
xprv9s21ZrQH143K2BEhMYpNQoUvAgiEjArAVaZaCTgsaGe6LsAnwubeiTcDzd23mAoyizm9cApe51gNfLMkBqkYoWWMCRwzfuJk8RwF1SVEpAQ

We now use the hd-private command to generate a hardened "account" key and a sequence of two private keys within the account:

$ bx hd-private --hard < master > account  
$ cat account  
xprv9vkDLt81dTKjwHB8fsVB5QK8cGnzveChzSrtCfvu3aMWvQaThp59ueufuyQ8Qi3qpjk4aKsbmbfxwcgS8PYbgoR2NWHeLyvg4DhoEE68A1n  
  
$ bx hd-private --index 0 < account  
xprv9xHfb6w1vX9xgZyPNXVgAhPxSsEkeRcPHEUV5iJcVEsuUEACvR3NRY3fpGhcnBiDbvG4LgndirDsia1e9F3DWPkX7Tp1V1u97HKG1FJwUpU  
  
$ bx hd-private --index 1 < account  
xprv9xHfb6w1vX9xjc8XbN4GN86jzNAZ6xHEqYxzbLB4fzHFd6VqCLPGRZFsdjsuMVERadbgDbziCRJru9n6tzEWrASVpEdrZrFidt1RDfn4yA3

Next, we use the hd-public command to generate the corresponding sequence of two public keys:

$ bx hd-public --index 0 < account  
xpub6BH1zcTuktiFu43rUZ2gXqLgzu5F3tLEeTQ5t6iE3aQtM2VMTxMcyLN9fYHiGhGpQe9QQYmqL2eYPFJ3vezHz5wzaSW4FiGrseNDR4LKqTy  
  
$ bx hd-public --index 1 < account  
xpub6BH1zcTuktiFx6CzhPbGjG3UYQ13WR16CmtbPiagEKpEVtpyjshWyMaMV1cn7nUPUkgQHPVXJVqsrA8xWbGQDhohEcDFTEYMvYzwRD7Juf8

The public keys can also be derived from their corresponding private keys using the hd-to-public command:

$ bx hd-private --index 0 < account | bx hd-to-public  
xpub6BH1zcTuktiFu43rUZ2gXqLgzu5F3tLEeTQ5t6iE3aQtM2VMTxMcyLN9fYHiGhGpQe9QQYmqL2eYPFJ3vezHz5wzaSW4FiGrseNDR4LKqTy  
  
$ bx hd-private --index 1 < account | bx hd-to-public  
xpub6BH1zcTuktiFx6CzhPbGjG3UYQ13WR16CmtbPiagEKpEVtpyjshWyMaMV1cn7nUPUkgQHPVXJVqsrA8xWbGQDhohEcDFTEYMvYzwRD7Juf8

We can generate a practically limitless number of keys in a deterministic chain, all derived from a single seed. This technique is used in many wallet applications to generate keys that can be backed up and restored with a single seed value. This is easier than having to back up the wallet with all its randomly generated keys every time a new key is created.

The seed can be encoded using the mnemonic-encode command:

$ bx hd-mnemonic < seed > words  
adore repeat vision worst especially veil inch woman cast recall dwell appreciate

The seed can then be decoded using the mnemonic-decode command:

$ bx mnemonic-decode < words  
eb68ee9f3df6bd4441a9feadec179ff1

Mnemonic encoding can make the seed easier to record and even remember.