

LIBRARY

A primer for libbitcoin

BY GROKCHAIN

READ IN 7 MIN

Introduction

In this tutorial, we will be taking a closer look at [libbitcoin](#), a multipurpose bitcoin library targeted at high-end use. An ideal backend to build fast implementations on top: mobile apps, desktop clients and server API's. The library places a heavy focus around asynchronicity, speed and availability.

Background

Originally started by [Amir Taaki](#) and now maintained by [Eric Voskuil](#), the libbitcoin toolkit is a set of cross platform C++ libraries for building bitcoin applications. The toolkit consists of several libraries, most of which depend on the foundational [libbitcoin](#) library.

Libraries

- [Bitcoin Blockchain Library](#)
- [Bitcoin High Performance Blockchain Database](#)
- [Bitcoin Cross-Platform C++ Development Toolkit](#)
- [Bitcoin Blockchain Query Protocol](#)
- [Bitcoin P2P Network Library](#)
- [Bitcoin Consensus Library](#)
- [Bitcoin Client Query Library](#)

Applications

Here are a list of applications built using the libbitcoin libraries.

- [Bitcoin Full Node](#)
- [Bitcoin Explorer Command Line Tool](#)
- [Bitcoin Full Node and Query Server](#)

In this tutorial, we will be looking at the applications, particularly using the [Bitcoin Explorer Command Line Tool](#) with our own [Bitcoin Full Node and Query Server](#) instance. Doing so will allow us to do exciting things like generating new addresses, or interacting with the bitcoin network.

```
gr0kchain $ bx seed | bx ec-new | bx ec-to-public | bx ec-to-address  
13ua8RRSxLpL5WL5cKUDePUcvJZgGWuKh7  
gr0kchain@bitcoinddev $ bx fetch-height tcp://mainnet2.libbitcoin.net:9091  
564292
```

Installing

The Bitcoin Explorer cli (`bx`) is a command line tool for working with Bitcoin. It can be built as a single portable executable for Linux, macOS or Windows and is available for download as a signed single executable for each. `bx` exposes over 80 commands and supports network communication with [Bitcoin Full Node and Query Server](#), and the P2P Bitcoin network. `bx` is well [documented](#) and supports simple and advanced scenarios, including stealth and multisig.

You can [install](#) your own copy of the libbitcoin server, or make use of the [available](#) community servers.

Let's get started

The `bx` cli includes various commands which can be categorised into:

- [Meta](#)
Invoke the help and settings
- [Wallet](#)
Bitcoin wallet related tools for private/public keys, wallet import formats, HD wallets, entropy for seed generation etc.

- **Stealth**
Tools for deriving stealth private/public keys, encoding and decoding addresses.
- **Messaging**
Signing and verification of messages.
- **Transaction**
Decode or encode transactions, scripts, converting scripts to addresses.
- **Online**
Communicate with libbitcoin servers.
- **Encoding**
Various encoding commands for addresses, base16, base58, base64 and base58check schemes.
- **Hashing**
Common bitcoin hashing functions for sha256, ripemd160.
- **Math**
Elliptic curve math functions for point addition, multiplication and conversion between BTC and satoshi's.

All the commands with exception to the `online` category can be executed without needing access to a libbitcoin server.

Configuration

Not all `bx` commands use configuration settings. However all commands process the configuration file if its path is specified.

The path to the configuration settings file is specified by the `--config` command line option, the `BX_CONFIG` environment variable, or by default as follows:

- Linux/OSX (prefix): `/etc/libbitcoin/bx.cfg`
- Linux/OSX (default): `/usr/local/etc/libbitcoin/bx.cfg`
- Windows: `%ProgramData%\libbitcoin\bx.cfg`
The Windows directory is hidden by default. If the specified file is not found default values are loaded. If the file contains invalid settings an error is returned via `STDERR`. If any setting is not specified its default is loaded.

For a list of defaults, execute the `settings` command from `bx`.

```
gr0kchain@bitcoind $ bx settings
settings
{
  network
  {
    channel_handshake_seconds 30
    connect_retries 0
    connect_timeout_seconds 5
    debug_file debug.log
    error_file error.log
    hosts_file hosts.cache
    identifier 3652501241
    seeds seed.bitcoin.sipa.be:8333,dnsseed.bluematt.me:8333,dnsseed.bitcoin.dashjr.org:8333,seed.bitcoinstats.
```

```
{
  client_private_key 0000000000000000000000000000000000000000000000000000000000000000
  connect_retries 0
  connect_timeout_seconds 5
  server_public_key 0000000000000000000000000000000000000000000000000000000000000000
  socks_proxy 0.0.0.0
  url tcp://mainnet.libbitcoin.net:9091
}
wallet
{
  hd_public_version 76067358
  hd_secret_version 76066276
  pay_to_public_key_hash_version 0
  pay_to_script_hash_version 5
  transaction_version 1
  wif_version 128
}
}
```

For convenience, a [bx.cfg](#) file is available for download which is populated with all default settings values that you can update depending on your requirements.

Let's take a look at some use cases.

Generating an address

Before generating an address, we need a good source of entropy for our private key

Note The details of elliptic curve math are not covered in this tutorial.

```
gr0kchain@bitcoind $ bx seed
bc22b367071fb6c72965b809cf2b11817e0eaea27d06523d
```

[Wikipedia]([https://en.wikipedia.org/wiki/Entropy_\(computing\)](https://en.wikipedia.org/wiki/Entropy_(computing))): In computing, entropy is the randomness collected by an operating system or application for use in cryptography or other uses that require random data. This randomness is often collected from hardware sources (variance in fan noise or HDD), either pre-existing ones such as mouse movements or specially provided randomness generators. A lack of entropy can have a negative impact on performance and security.

WARNING: Pseudorandom seeding can introduce cryptographic weakness into your keys. This `seed` command is provided as a convenience and not recommended for creating secure private keys.

Now we can generate a private key using our source of entropy.

```
gr0kchain@bitcoind $ bx ec-new bc22b367071fb6c72965b809cf2b11817e0eaea27d06523d
c93fc696e57e69462dfc13b3986be443f470ad8ad8b2bb084c31d1f874a0e934
```

Bitcoin Developer Network – A primer for libbitcoin

[Share this](#)

```
gr0kchain@bitcoindev $ bx ec-to-public c93fc696e57e69462dfc13b3986be443f470ad8ad8b2bb084c31d1f874a0e934
038f0c6d3100303b560cb718bdc4a8224a866e9a264cc99ffa5bee6ac83d103d24
```

Defaults to the compressed public key format

Generate an address from the public key

```
gr0kchain@bitcoindev $ bx ec-to-address 038f0c6d3100303b560cb718bdc4a8224a866e9a264cc99ffa5bee6ac83d103d24
1ERgdZfHU1MZyn6BgyVEGSbBgbzTa39uQa
```

The `bx` command also follows the unix philosophy of minimalism and modularity, by doing one thing, and doing it well. This allows you to combined into a single command, kind of like the Swiss Army knife for bitcoin really.

```
gr0kchain@bitcoindev $ bx seed | bx ec-new | bx ec-to-public | bx ec-to-address
1AoPnzTFUr6EoyNTsHWi2nqRZxgtJx89RD
```

Convert these into QRcodes.

```
gr0kchain@bitcoindev $ bx qrcode -p 131zKT2n1FN4Z6JdDAWMg3w8ehYjoRByTB | imgcat
```



You might wish to output the `qrcode` data to file. Here we are using imgcat, it's like `cat`, but for images.

We can also generate a BIP39 mnemonic phrase (much simpler for humans to understand).

```
gr0kchain@bitcoindev $ bx seed | bx mnemonic-new
shove morning usage area fox knee diary else explain blush pupil awful funny sphere tower rose budget half
```

Increasing the security of our seed is also an option.

```
gr0kchain@bitcoindev $ bx seed -b 256 | bx mnemonic-new
wait chuckle orchard digital section seminar loud dust dog very exclude lift resource clinic between soup match sta
```

```
gr0kchain@bitcoindev $ bitcoin-cli sendtoaddress 2N1Bnuqq2xSKNPTVrpvSmmE2MFdo9KFo3JD 1
97d7f218a29f46521920d3b397ac1c0fbc97b56d68442e45b74f19edc77f23bb
gr0kchain@bitcoindev $ bitcoin-cli getrawtransaction 97d7f218a29f46521920d3b397ac1c0fbc97b56d68442e45b74f19edc77f23
transaction
{
  hash 97d7f218a29f46521920d3b397ac1c0fbc97b56d68442e45b74f19edc77f23bb
  inputs
  {
    input
    {
      previous_output
      {
        hash 5f2ef5c049609aa9d4391cfd76a56af4e3924b9f0b3246ed79b8d2adaf5ceeea
        index 0
      }
      script [304402202cc57b57ae5d80379556a9e4d0e84797d366e43260d57e73a8318fa665304e3d02205e099f7ac7e19b30083
        sequence 4294967294
      ]
    }
  }
  lock_time 104
  outputs
  {
    output
    {
      address_hash d48a1f5c06037474c15e6775f82d99a673af6059
      script "hash160 [d48a1f5c06037474c15e6775f82d99a673af6059] equal"
      value 4899996260
    }
    output
    {
      address_hash 571ad354f9733db9f42ed3a1ee5f5a54e91c13c6
      script "hash160 [571ad354f9733db9f42ed3a1ee5f5a54e91c13c6] equal"
      value 100000000
    }
  }
  version 2
}
```

Communicating with a libbitcoin-server

So far, we've only looked at commands available from `bx` that can be executed offline. There are however times where it might be useful to interact with the bitcoin network.

Configuration and setup

The default settings are close to optimal for initial block downloading the `mainnet` chain data. Once complete, settings should be changed to recommended operational configuration.

For learning purposes however, we usually recommend using a bitcoin instance configured in [regtest](#) mode.

```
gr0kchain@bitcoindev $ mkdir ~/.bs/ && cd ~/.bs/
gr0kchain@bitcoindev $ cp ~/path/to/regtest.conf ~/.bs/regtest.conf
gr0kchain@bitcoindev $
[network]
# The magic number for message headers.
identifier = 3669344250

# The port for incoming connections.
inbound_port = 18445

# Disable seeding, otherwise manually populate hosts.cache file.
host_pool_capacity = 0

# Optionally connect to one or more peers on a private regtest network.
peer = localhost:18444

[blockchain]
# Disable default (mainnet) checkpoints.
checkpoint = 0f9188f13cb7b2c71f2a335e3a4fc328bf5beb436012afca590b1a11466e2206:0

[fork]
# Retarget difficulty, defaults to true (use false for regtest).
retarget = false

# Require coinbase input includes block height (disabled on satoshi client).
bip34 = false
gr0kchain@bitcoindev $ cd bs ~/.bs/regtest.conf --initchain
09:46:56.011445 INFO [server] ===== startup =====
09:46:56.011480 WARNING [server] ===== startup =====
09:46:56.011631 ERROR [server] ===== startup =====
09:46:56.011667 FATAL [server] ===== startup =====
09:46:56.011692 INFO [server] Using config file: "/home/gr0kchain/.bs/regtest.conf"
09:46:56.011771 INFO [server] Please wait while initializing "blockchain" directory...
09:47:01.622075 INFO [server] Completed initialization.
gr0kchain@bitcoindev $ bs -c ~/.bs/regtest.conf

gr0kchain@bitcoindev $ bitcoin-cli --version
Bitcoin Core RPC client version v0.15.2
gr0kchain@bitcoindev $ bitcoin-cli generate 101
[
  "150270ef73b5dbbecb8232fb110673972e5fad138b067318d28170c83ab350bf"
]
```

You might wish to consult this [github issue](#) for more information on why versions prior to Bitcoin Core 0.16 don't work.

Examples

Here are some simple examples demonstrating the use of `online` commands using `bx`.

```
gr0kchain@bitcoind $ bx fetch-height tcp://localhost:9091
101
```

Another is to return the block header for block 100.

```
gr0kchain@bitcoind $ bx fetch-header --height 100
header
{
  bits 545259519
  hash 2e89935f5c333d90a771ca7a96babbdad06a0f106e9dfd7c17173b93d66a8d
  merkle_tree_hash d75abaa8bc2cd1ae4021da8a239035316cd5e89d93e72f7f3e354b3150e1aa65
  nonce 1
  previous_block_hash 6d77c75e8b420330bb23a98c4c03c6e39773157e6b22ddcf7396b52cbff5d0b5
  time_stamp 1550928703
  version 536870912
}
```

To query your local node, update your local `bx` config by setting the `url` to your local libbitcoin-server.

```
[server]
url = tcp://localhost:9091
```

More information

The libbitcoin `bx` cli is packed with many features. Feel free to explore further by invoking the `help` meta command.

```
gr0kchain@bitcoind $ bx help
```

```
Usage: bx COMMAND [--help]
```

```
Version: 3.2.0
```

```
Info: The bx commands are:
```

```
address-decode
address-embed
address-encode
base16-decode
base16-encode
base58-decode
base58-encode
base58check-decode
base58check-encode
base64-decode
base64-encode
bitcoin160
bitcoin256
```


Bitcoin Developer Network – A primer for libbitcoin

[Share this](#) 

cert-public
ec-add
ec-add-secrets
ec-multiply
ec-multiply-secrets
ec-new
ec-to-address
ec-to-ek
ec-to-public
ec-to-wif
ek-address
ek-new
ek-public
ek-public-to-address
ek-public-to-ec
ek-to-address
ek-to-ec
fetch-balance
fetch-header
fetch-height
fetch-history
fetch-public-key
fetch-stealth
fetch-tx
fetch-tx-index
fetch-utxo
hd-new
hd-private
hd-public
hd-to-ec
hd-to-public
help
input-set
input-sign
input-validate
message-sign
message-validate
mnemonic-new
mnemonic-to-seed
qrcode
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-public
stealth-secret
stealth-shared
token-new
tx-decode
tx-encode
tx-sign
uri-decode
uri-encode
validate-tx
watch-address
watch-stealth
watch-tx
wif-to-ec
wif-to-public
wrap-decode
wrap-encode
```

Bitcoin Explorer home page:

<https://github.com/libbitcoin/libbitcoin-explorer>

You can also get help for a specific command by passing it as the first parameter to the help command.

```
gr0kchain@bitcoind $ bx help seed
```

Usage: **bx** seed [-h] [--bit_length value] [--config value]

Info: Generate a pseudorandom seed.

Options (named):

```
-b [--bit_length]    The length of the seed in bits. Must be divisible by
                    8 and must not be less than 128, defaults to 192.
-c [--config]        The path to the configuration settings file.
-h [--help]          Get a description and instructions for this command.
```

Conclusion

In closing, we have looked into some of the useful things we can do with the `libbitcoin` applications. We looked into using the `bx` command for generating addresses, as well as setting up and configuring our own local `libbitcoin-server`.

Reference

[Libbitcoin Bitcoin Explorer](#)

[Libbitcoin Server](#)

Bitcoin Developer Network – A primer for libbitcoin

Share this 

17



SHARE



WRITTEN BY

**GROKCHAIN**Digital Anarchist accepting Bitcoin donations at | bc1qf2kn6vyq5995uhjkhpxd9q7tave703g0r07fx4

📍 127.0.0.1

[HTTPS://BITCOINDEV.NETWORK](https://bitcoinddev.network)[TWITTER](#)

COMMENTS

No Comments Yet



SUBSCRIBE TO BITCOIN DEVELOPER NETWORK WEEKLY NEWSLETTER

Enter your email address to subscribe to this blog and receive notifications of new posts by email.

NEXT

Money as a content type

FEBRUARY 25, 2019

PREVIOUS

Accessing Bitcoin's ZeroMQ interface

FEBRUARY 19, 2019

LATEST POSTS

TAGS

BITCOIN DEVELOPER
NETWORK ON
@INSTAGRAM

**Bitcoin wire
protocol 101**

JANUARY 27, 2020

**Introduction to
Bitcoin
Programming with
BitcoinJS and Bitcoin Core**

DECEMBER 19, 2019

- BLOCKCHAIN
- CLI
- COMMUNITY
- CRYPTOGRAPHY
- DATA
- DNS
- ENVIRONMENT
- EXPLOIT
- HASH FUNCTIONS
- JAVASCRIPT
- JULIA
- LEVELDB
- LIBRARY
- LIGHTNING
- LIQUID
- MEMPOOL
- MERKLE ROOT
- MERKLE TREES
- MUSIG
- NAS
- NETWORKING
- NEWSLETTER
- NODE
- PGP
- PHP
- PROTOCOL
- REGTEST
- RPC
- SANDBOX
- SCHNORR
- SCRIPT
- SECP256K1
- SECURITY
- SERIES
- SHELL
- STANDARDS
- STATISTICS
- SYNOLOGY
- TRANSACTIONS



