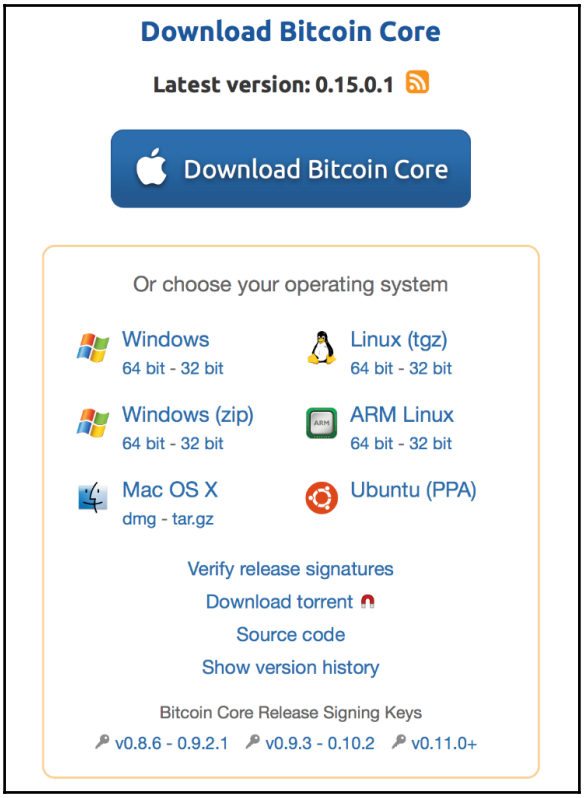
7
Bitcoin Clients and APIs

In this chapter, we provides you with an introduction to Bitcoin client installation and a basic introduction to various APIs and tools that are available for developing Bitcoin applications. We will examine how to set up a Bitcoin node in live and test networks. Also, we will discuss various commands and utilities that are used to perform various functions in Bitcoin system.

Bitcoin installation

The Bitcoin Core client can be installed from `https://bitcoin.org/en/download`. This is available for different architectures and platforms ranging from x86 Windows to ARM Linux, as shown in the following screenshot:



Download Bitcoin Core

Types of Bitcoin Core clients

Let's explore the different types of Bitcoin Core clients.

Bitcoind

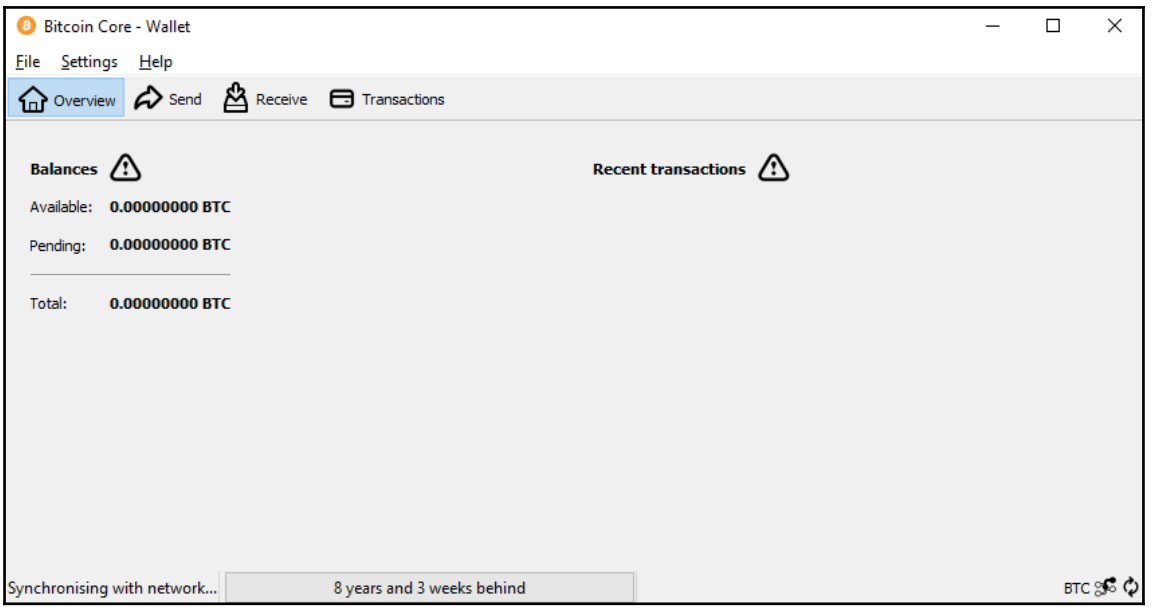
This is the core client software that can be run as a daemon, and it provides the JSON RPC interface.

Bitcoin-cli

This is the command line feature-rich tool to interact with the daemon; the daemon then interacts with the blockchain and performs various functions. Bitcoin-cli calls only JSON-RPC functions and does not perform any actions on its own on the blockchain.

Bitcoin-qt

This is the Bitcoin Core client GUI. When the wallet software starts up first, it verifies the blocks on the disk and then starts up and shows the following GUI:

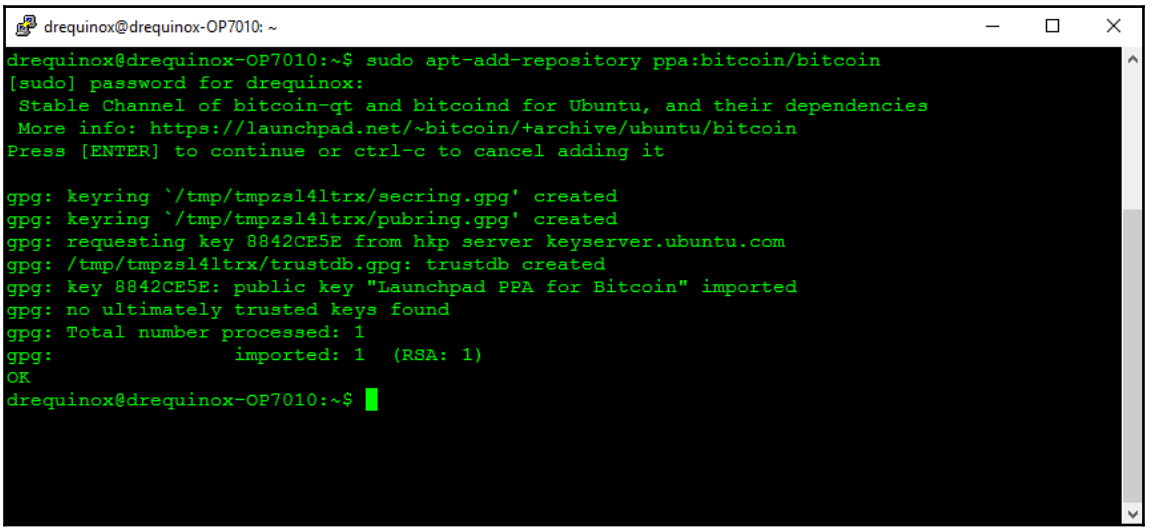


Bitcoin Core QT client, just after installation, showing that blockchain is not in sync

The verification process is not specific to the Bitcoin-qt client; it is performed by the Bitcoind client as well.

Setting up a Bitcoin node

A sample run of the Bitcoin Core installation on Ubuntu is shown here; for other platforms, you can get details from `https://bitcoin.org/en/`:



Bitcoin setup

1. Run the following command:

```
$ sudo apt-get update
```

2. Depending on the client required to be installed, users can use either of the following commands, or they can issue both commands at once:

```
$ sudo apt-get install bitcoind
$ sudo apt-get install bitcoin-qt
$ sudo apt-get install bitcoin-qt bitcoind
Reading package lists... Done
Building dependency tree
Reading state information... Done
.......
```

Setting up the source code

The Bitcoin source code can be downloaded and compiled if users wish to participate in the Bitcoin code or for learning purpose. The `git` command can be used to download the Bitcoin source code:

```
$ sudo apt-get install git
$ mkdir bcsource
$ cd bcsource
$ git clone https://github.com/bitcoin/bitcoin.git
Cloning into 'bitcoin'...
remote: Counting objects: 78960, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 78960 (delta 0), reused 0 (delta 0), pack-reused 78957
Receiving objects: 100% (78960/78960), 72.53 MiB | 1.85 MiB/s, done.
Resolving deltas: 100% (57908/57908), done.
Checking connectivity... done.
```

Change the directory to `bitcoin`:

```
$ cd bitcoin
```

After the preceding steps are completed, the code can be compiled:

```
$ ./autogen.sh
$ ./configure.sh
$ make
$ sudo make install
```

Setting up bitcoin.conf

The `bitcoin.conf` file is a configuration file that is used by the Bitcoin Core client to save configuration settings. All command-line options for the `bitcoind` client with the exception of the `-conf` switch can be set up in the configuration file, and when Bitcoin-qt or Bitcoind will start up, it will take the configuration information from that file.

In Linux systems, this is usually found in `$HOME/.bitcoin/`, or it can also specified in the command line using the `-conf=<file>` switch to Bitcoind core client software.

Starting up a node in testnet

The bitcoin node can be started in the testnet mode if you want to test the Bitcoin network and run an experiment. This is a faster network as compared to the live network and has relaxed rules for mining and transactions.

Various faucet services are available for the bitcoin test network. One example is Bitcoin TestNet sandbox, where users can request bitcoins to be paid to their testnet bitcoin address.

This can be accessed via `https://testnet.manu.backend.hamburg/`.

This is very useful for experimentation with transactions on testnet.

The command line to start up testnet is as follows:

```
bitcoind --testnet -daemon
bitcoin-cli --testnet <command>
bitcoin-qt --testnet
```

Starting up a node in regtest

The regtest mode (regression testing mode) can be used to create a local blockchain for testing purposes.

The following commands can be used to start up a node in the regtest mode:

```
$ bitcoind -regtest -daemon
Bitcoin server starting
```

Blocks can be generated using the following command:

```
$ bitcoin-cli -regtest generate 200
```

Relevant log messages can be viewed in the `.bitcoin/regtest` directory on a Linux system under `debug.log`:

```
drequinox@drequinox-OP7010:~/.bitcoin/regtest$ tail -f debug.log
2016-10-16 15:43:55 AddToWallet d461e1fb162dd6958139a2ab5e4f9993ffbd51b1a4e3a80e5b77e472cd90dd6a  new
2016-10-16 15:43:55 CreateNewBlock(): total size 1000 txs: 0 fees: 0 sigops 400
2016-10-16 15:43:55 UpdateTip: new best=37c1f40299a3724dd2edf63d26925cb580b8c5f27405289ef9204e53fe4e1b87 height=299 version=0x30000003 log2_work=9.22881
87 tx=300 date='2016-10-16 15:44:27' progress=1.000000 cache=0.1MiB(299tx)
2016-10-16 15:43:55 AddToWallet b88883e122c4f3ae66b53e4026d8fa6c916c570df2b154febf51c676235d70bf  new
2016-10-16 15:43:55 CreateNewBlock(): total size 1000 txs: 0 fees: 0 sigops 400
2016-10-16 15:43:55 UpdateTip: new best=5c22d0b090b6f3fd978fbbb14803d1d34eccccfb697a199d502beb1d88da43ad2 height=300 version=0x30000003 log2_work=9.23361
97 tx=301 date='2016-10-16 15:44:28' progress=1.000000 cache=0.1MiB(300tx)
2016-10-16 15:43:55 AddToWallet e315c5b6863aed2d4477f6e6e5cdb7ace273f40549d249b90b8793de0de0b8e1  new
2016-10-16 15:43:55 CreateNewBlock(): total size 1000 txs: 0 fees: 0 sigops 400
2016-10-16 15:43:55 UpdateTip: new best=7f9eeb78cdb34f374d426c95aab82c85810715574c0a87ec93218ab77ae9f5ae height=301 version=0x30000003 log2_work=9.23840
47 tx=302 date='2016-10-16 15:44:28' progress=1.000000 cache=0.1MiB(301tx)
2016-10-16 15:43:55 AddToWallet 428058e9e73f6862f8e126999efa4062dad2e63b253630d2e2ec086e7f5ac029  new
```

Messages in Bitcoin debug log

After block generation, the balance can be viewed as follows:

```
$ bitcoin-cli -regtest getbalance
8750.00000000
```

The node can be stopped using this:

```
$ bitcoin-cli -regtest stop
Bitcoin server stopping
```

Experimenting with Bitcoin-cli

Bitcoin-cli is the command-line interface available with the Bitcoin Core client and can be used to perform various functions using the RPC interface provided by the Bitcoin Core client:

```
drequinox@drequinox-OP7010:~$ bitcoin-cli getinfo
{
  "version": 130000,
  "protocolversion": 70014,
  "walletversion": 130000,
  "balance": 0.00000000,
  "blocks": 433948,
  "timeoffset": 0,
  "connections": 8,
  "proxy": "",
  "difficulty": 258522748404.5154,
  "testnet": false,
  "keypoololdest": 1475534258,
  "keypoolsize": 100,
  "paytxfee": 0.00000000,
  "relayfee": 0.00001000,
  "errors": ""
}
drequinox@drequinox-OP7010:~$
```

A sample run of bitcoin-cli getinfo; the same format can be used to invoke other commands

A list of all commands can be shown via the command shown in the following screenshot:

```
drequinox@drequinox-OP7010:~$ bitcoin-cli -testnet help | more
== Blockchain ==
getbestblockhash
getblock "hash" ( verbose )
getblockchaininfo
getblockcount
getblockhash index
getblockheader "hash" ( verbose )
getchaintips
getdifficulty
getmempoolancestors txid (verbose)
getmempooldescendants txid (verbose)
getmempoolentry txid
getmempoolinfo
getrawmempool ( verbose )
gettxout "txid" n ( includemempool )
gettxoutproof ["txid",...] ( blockhash )
gettxoutsetinfo
verifychain ( checklevel numblocks )
verifytxoutproof "proof"

== Control ==
getinfo
help ( "command" )
stop
```

Testnet bitcoin-cli' this is just the first few lines of the output, actual output has many commands

The preceding screenshot shows a list of various command-line options available in Bitcoin-cli, the Bitcoin command-line interface. These commands can be used to query the blockchain and control the local node.

Starting from Bitcoin Core client 0.10.0, the HTTP REST interface is also available. By default, this runs on the same TCP port `8332` as JSON-RPC.

Bitcoin programming and the command-line interface

Bitcoin programming is a very rich field now. The Bitcoin Core client exposes various JSON RPC commands that can be used to construct raw transactions and perform other functions via custom scripts or programs. Also, the command-line tool, Bitcoin-cli, is available, which makes use of the JSON-RPC interface and provides a rich toolset to work with Bitcoin.

These APIs are also available via many online service providers in the form of bitcoin APIs, and they provide a simple HTTP REST interface. Bitcoin APIs, such as Blockchain.info (`https://blockchain.info/api`) and BitPay (`https://bitpay.com/api`), Block.io (`https://www.block.io`), and many others, offer a myriad of options to develop Bitcoin-based solutions.

Various libraries are available for bitcoin programming. A list is shown as follows, and those of you interested can further explore the libraries.

- **Libbitcoin**: Available at `https://libbitcoin.dyne.org/` and provides powerful command-line utilities and clients
- **Pycoin**: Available at `https://github.com/richardkiss/pycoin`, is a library for Python
- **Bitcoinj**: This library is available at `https://bitcoinj.github.io/` and is implemented in Java

There are many online bitcoin APIs available; the most commonly used APIs are listed as follows:

- `https://bitcore.io/`
- `https://bitcoinjs.org/`
- `https://blockchain.info/api`

As all APIs offer almost similar type of functionality, it can get confusing to decide which one to use. It is also difficult to recommend which API is the best because all APIs are similarly feature-rich. One thing to keep in mind, however, is, security, so whenever you evaluate an API for usage, in addition to assessing the offered features, also evaluate how secure is the design of the API.

Summary

We began this chapter with the introduction of Bitcoin installation, followed by some discussion on source code setup and setup of Bitcoin clients for various networks. After this, we examined various command-line options available in the Bitcoin client.

Finally, we also saw which APIs are available for Bitcoin programming and main points to keep in mind while evaluating APIs for usage.

In the next chapter, we will explore alternative coins that emerged after bitcoin. We will also examine in detail various properties and attributes associated with the alternative cryptocurrencies.