Product Document: XP Coin Product Building Environment

Author: DigiWarfare/XPCoin, Tnaruto/XPCoinJP, Lexicon/XPCoin

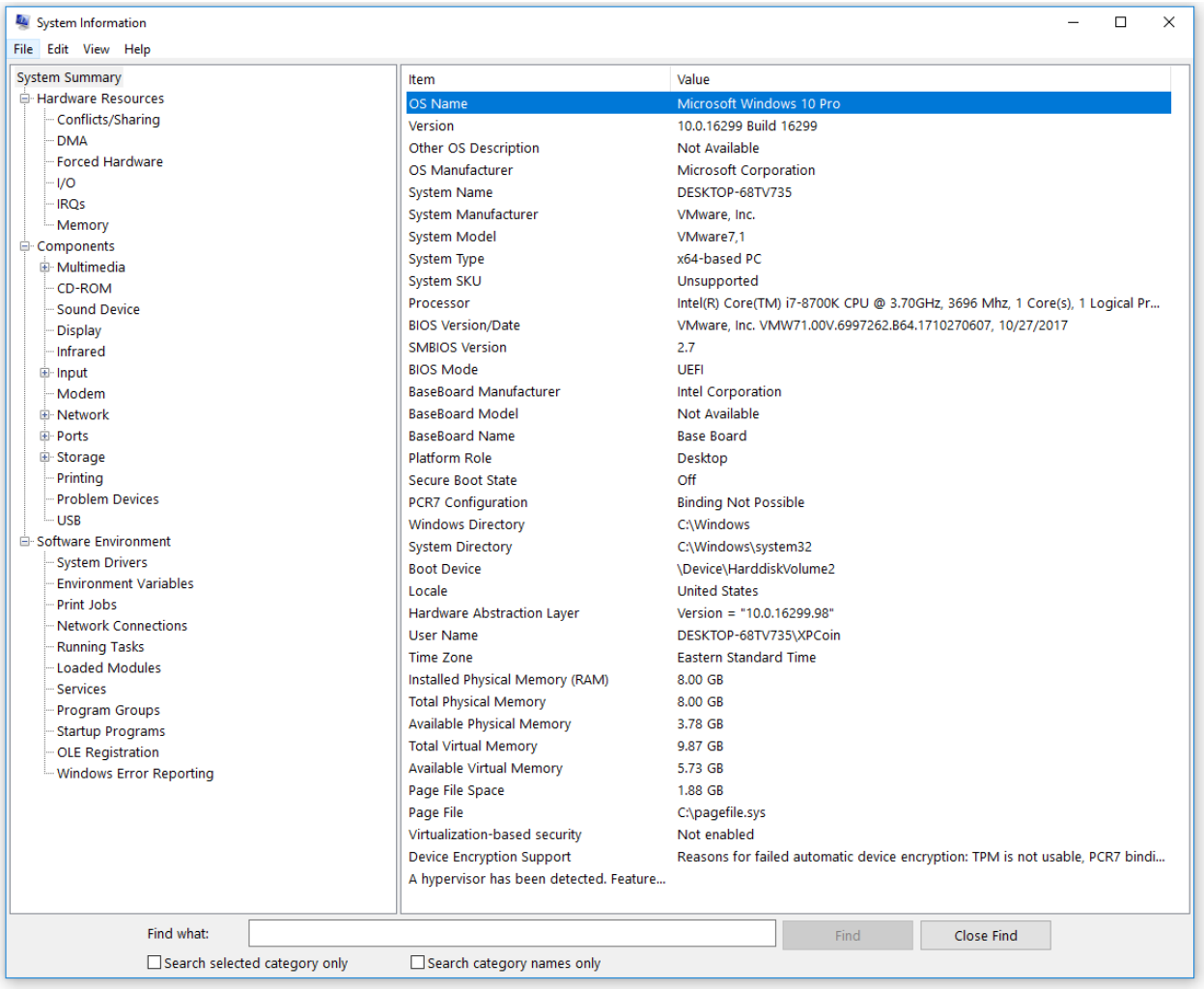
Version 1.0.2

March 6, 2018

This document covers building of the Windows 10, Mac OS X, and Ubuntu Linux wallets. It gives detail and setup instructions to guide you and assist you in any issues at you may encounter. This document will be revised periodically to contain updated to the environment and to the XPCoin software itself.

Part 1: Building the windows wallet

This section will cover building the windows wallets. This will cover both the server daemon and the qt wallet version.



Pathing Requirements:

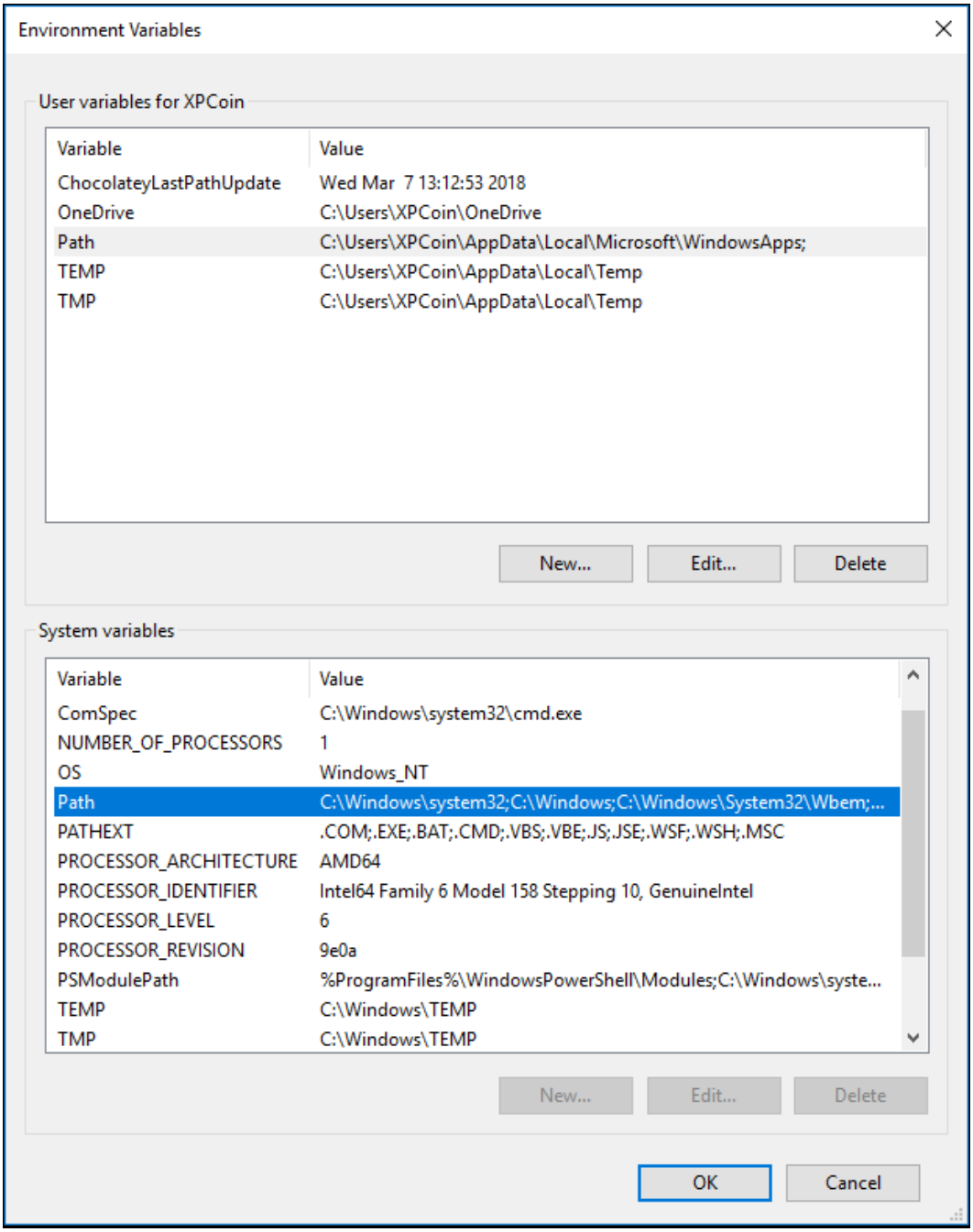
PATH is an environment variable on Unix-like operating systems, DOS, OS/2, and Microsoft Windows, specifying a set of directories where executable programs are located. In general, each executing process or user session has its own PATH setting.

You will need to make sure these following items are added to the windows path.

1. CMake is an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files and generate native makefiles and workspaces that can be used in the compiler environment of your choice. The suite of CMake tools were created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects such as ITK and VTK.

a. C:\Program Files\CMake\bin

2. devenv.exe is a process which belongs to the Microsoft Visual Studio, which is an application development suite from Microsoft. This program is a non-essential system process but should not be terminated unless suspected to be causing problems.
    a. C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\IDE



Requirements:

Building the wallet will require you to have the following software packages. Please make sure that you have before compiling or running the batch files.

1. Windows 10 Pro
2. Visual Studio 2017 Community
    a. https://www.visualstudio.com/thank-you-downloading-visual-studio/?sku=Community&rel=15
3. Chocolatey
    a. https://chocolatey.org/
4. Qt 5.10.1: Open Source (Community Version)
    a. https://www.qt.io/download
    b. Install to HD location **C:\QT**
5. Visual C++ Redistributable for Visual Studio 2015 (msvc2015 32bit)
    a. https://www.microsoft.com/en-us/download/details.aspx?id=48145
6. Windows SDK 8.1 and Windows 10 SDK (ver. 10.0.16299.91)
    a. https://developer.microsoft.com/en-us/windows/downloads/sdk-archive
7. GitHub CLI
    a. https://git-scm.com/download/win

Getting the XPCoin Software:

The next steps will have to be performed in a command shell being run as administrator

1. XPCoin GitHub: https://github.com/eXperiencePoints/XPCoin
    a. git clone https://github.com/eXperiencePoints/XPCoin
2. Add features required for build environment
    a. cd XPCoin
    b. git checkout feature/support_windows
    c. git checkout feature/add_appveyor_script

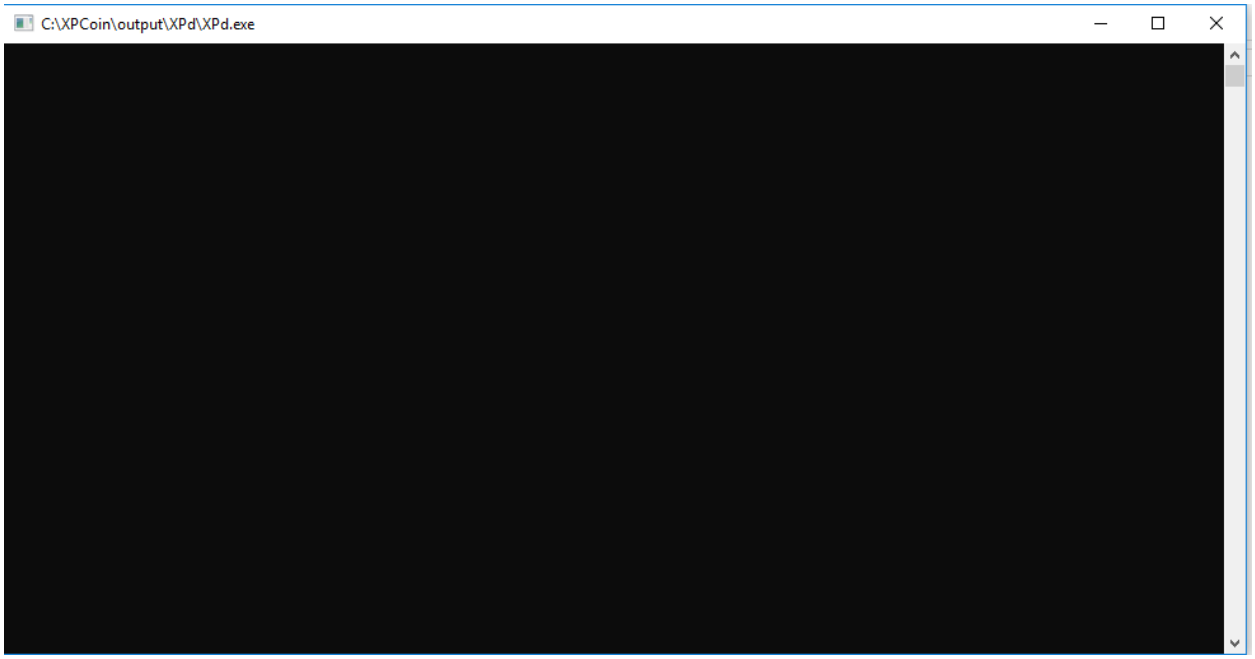
3. Installing Dependency's: We need to install a number of build dependency's. Please execute the next steps in exact order. If a step fails to build, please correct it before moving on. You can however start over and run all files in order. This will not affect the system as each part is separate or requires a previous to be built before it can. These files need to be run from a command shell as administrator.

    a. execute MSVC\prepare.bat by Admin permission.
    b. execute MSVC\boost.bat that downloads precompiled boost 1.66.0
    c. execute MSVC\openssl.bat that downloads precompiled openssl 1.0.2l
    d. execute MSVC\berkeleydb.bat that downloads berkeley db 4.8.30 sources.
    e. build berkeley db by using MSVC\db-4.8.30.NC\build_windows\Berkeley_DB.sln.
        i. After opening solution, go to project menu and chose retarget solution. Choose 8.1 and retarget. After this, choose retarget solution again and now choose windows 10. Once this has been done, Berkeley db will build successfully.
    f. execute MSVC\qrencode.bat that downloads and compiles libqrencode. (for qt)

XPd: Headless Server Wallet

XPd is the windows server binary. It is meant to run and produce no output to the user, as it meant to be accessed through RPC by applications.

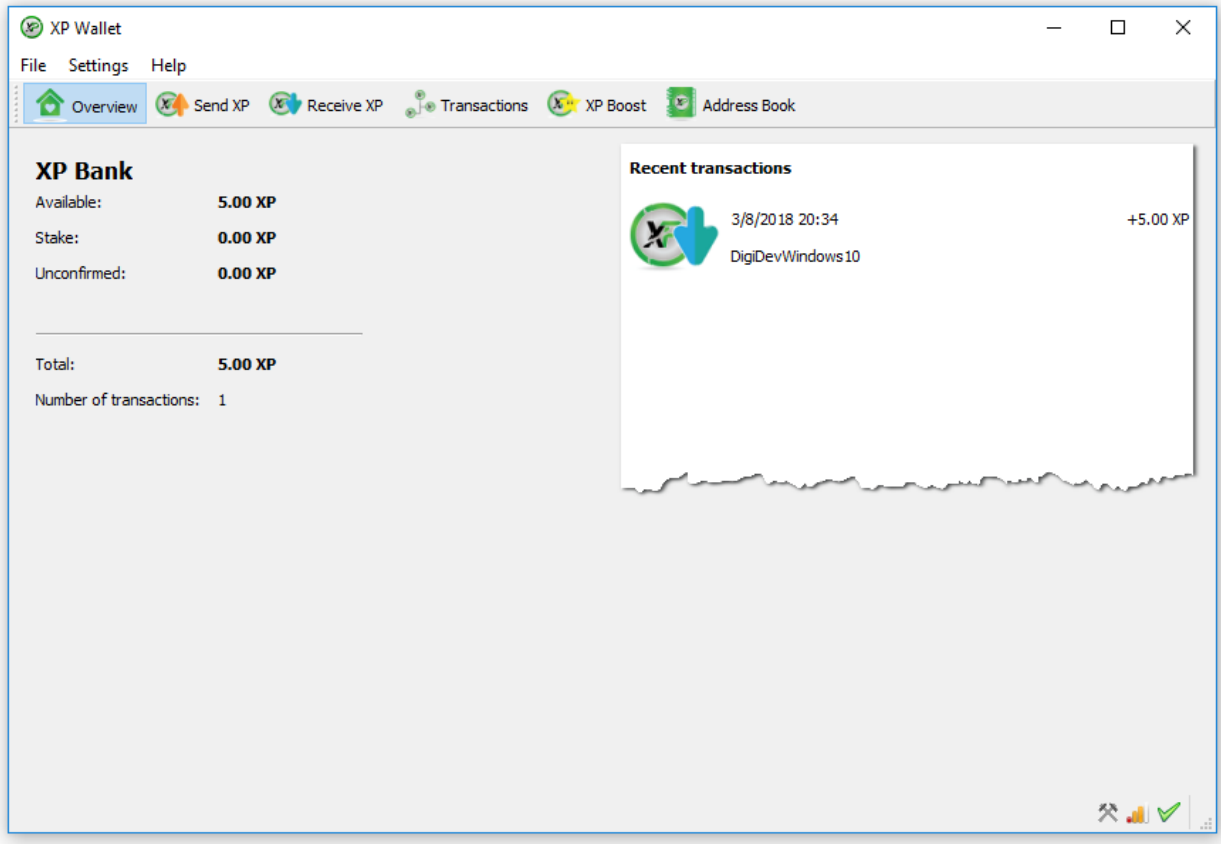
1. execute MSVC\XPd.bat that compiles XPd.exe.
    a. This file needs to be run from a command shell as administrator.
2. C:\XPCoin\output\XPd\XPd.exe
3. Debug Log: C:\Users\XPCoin\AppData\Roaming\XP\debug.log
4. Config File: C:\Users\XPCoin\AppData\Roaming\XP\XP.conf
5. Peer File: C:\Users\XPCoin\AppData\Roaming\XP\peers.dat
6. Wallet File: C:\Users\XPCoin\AppData\Roaming\XP\wallet.dat



XP-Qt: GUI Desktop Wallet

XP-QT is the windows user desktop wallet. This wallet has a full GUI and can be used to stake.

1. execute MSVC\XP-Qt.bat
    a. This file needs to be run from a command shell as administrator.
2. C:\XPCoin\output\XPd\ XP-Qt.exe
3. Debug Log: C:\Users\XPCoin\AppData\Roaming\XP\debug.log
4. Config File: C:\Users\XPCoin\AppData\Roaming\XP\XP.conf
5. Peer File: C:\Users\XPCoin\AppData\Roaming\XP\peers.dat
6. Wallet File: C:\Users\XPCoin\AppData\Roaming\XP\wallet.dat



Part 2: MAC OS X: High Serra 10.13.3 DMG QT Build

Please make sure that your mac is fully patched and can be connected to iTunes. We will need to install software such as XCode and qt creator**.**

Requirements:

1. MAC OS X High Serra
    a. Version **10.13.3**
2. Install XCODE 8.1
    a. https://itunes.apple.com/us/app/xcode/id497799835?mt=12
3. Install Qt Creator 5
4. Install GIT
    a. http://mac.github.com/
5. Install Brew
    a. /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

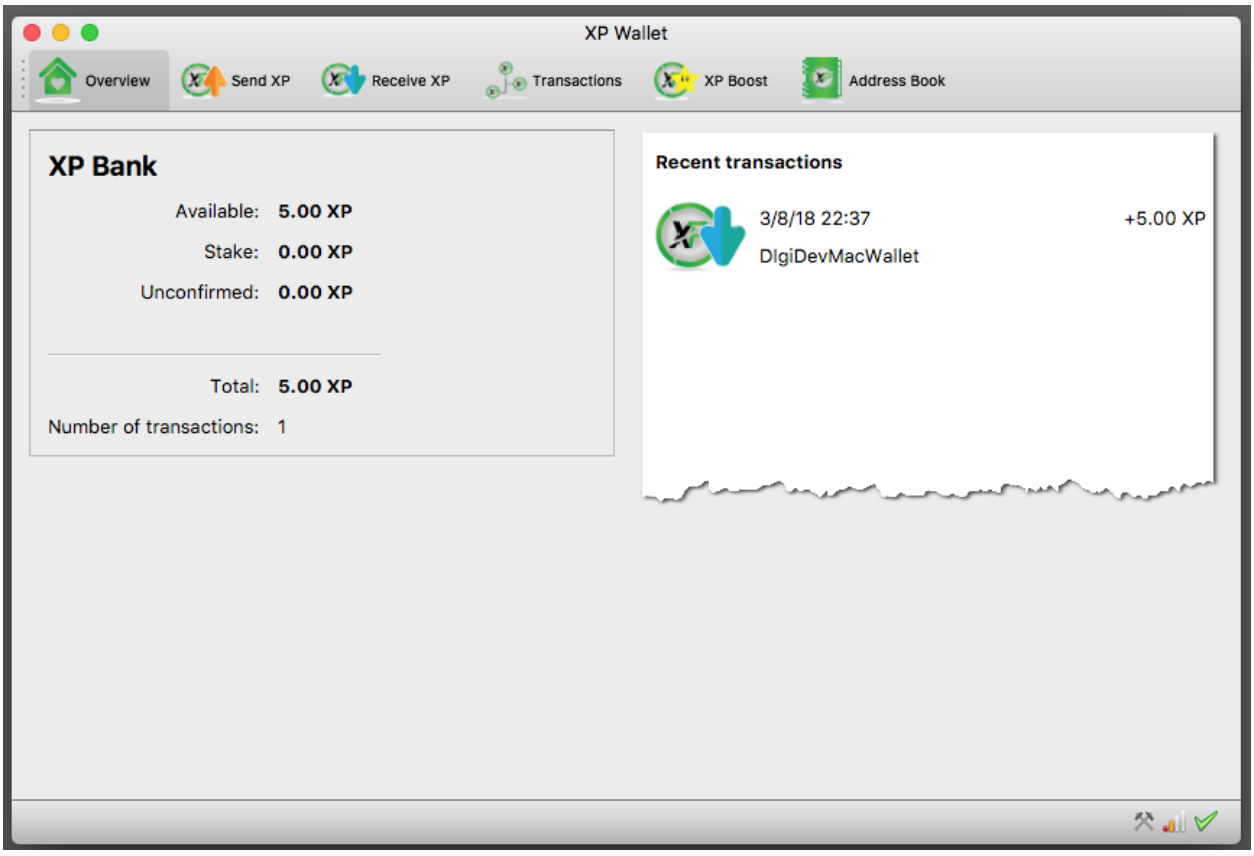
Mac OS X QT Wallet Build Instructions:

The following will install the dependency's that are needed to build the wallet. Please run them in order and make sure that each step is completed before moving on to the next.

1. sudo xcode-select –install
2. brew install autoconf automake libtool boost openssl pkg-config protobuf qt qrencode berkeley-db@4
3. git clone https://github.com/eXperiencePoints/XPCoin
4. cd XPCoin
5. git checkout feature/add_appveyor_script

Your environment should now be set up and ready to compile the XPCoin QT Mac Wallet.

6. `brew --prefix qt`/bin/qmake
7. make
8. `brew --prefix qt`/bin/macdeployqt XP-Qt.app -dmg



Part 3: Debian and Ubuntu 16.04 Desktop

Requirements:

Make sure system is fully patched before trying to build the ubuntu wallet.

1. Ubuntu 16.04 and64 Desktop
    a. http://releases.ubuntu.com/16.04/ubuntu-16.04.4-desktop-amd64.iso
    b. 4gig of memory
    c. 25 gig drive space

Ubuntu QT Wallet Build Instructions:

First, make sure that the required packages for Qt5 development of your distribution are installed, these are

1. sudo add-apt-repository ppa:bitcoin/bitcoin -y
2. sudo apt-get update
3. sudo apt-get install git build-essential libssl-dev libdb4.8-dev libdb4.8++-dev \
   libboost-all-dev libqrencode-dev libminiupnpc-dev qt5-default qttools5-dev-tools

Source Code: Get the wallet code from GitHub

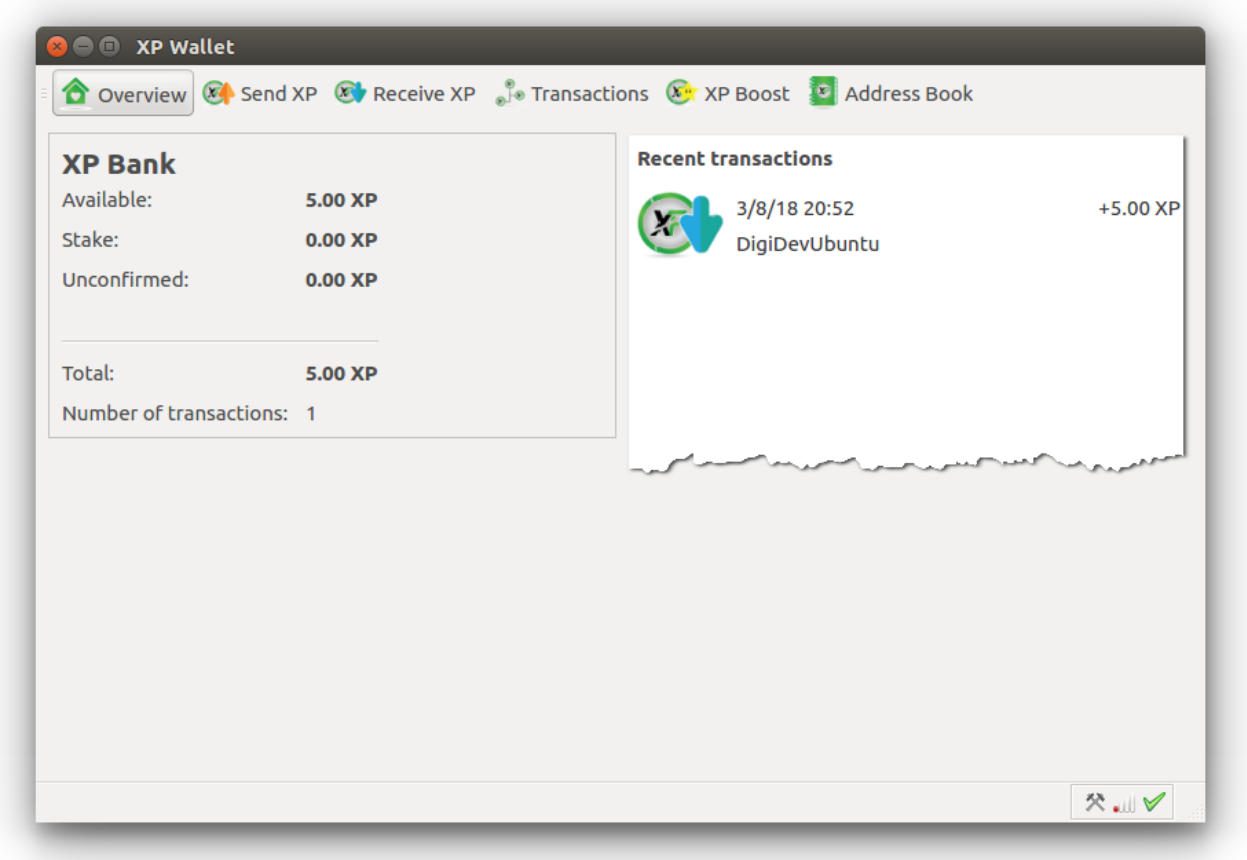
1. git clone https://github.com/eXperiencePoints/XPCoin
2. cd XPCoin
3. git checkout feature/add_appveyor_script

then execute the following two commands:

1. qmake
2. make

Alternatively, install Qt Creator and open the XP-qt.pro file.

An executable named XP-qt will be built.



If you wish to build the headless wallet you can do the following commands from inside the XPCoin/src folder. This will produce XPd in the same directory.

1 make -f makefile.unix

Part 4: Debug Log File:

The debug file is created when the application is running. Please use the debug to log any errors and please pay attention to it as it contains a lot of useful information. It will be located in the following folder C:\Users\XPCoin\AppData\Roaming\XP\ in this build example.

As the application runs this file will grow in size. Periodically removing the file will make troubleshooting easier. If deleted this file will regenerate on the next application start.

Debug Log File Contents Example:

```
XP version XP-v1.1.0.2-bdb-msvc (2018-03-07 22:01:00 +0900)
Using OpenSSL version OpenSSL 1.0.2l  25 May 2017
Startup time: 03/09/18 01:12:03
Default data directory C:\Users\XPCoin\AppData\Roaming\XP
Used data directory C:\Users\XPCoin\AppData\Roaming\XP
dbenv.open LogDir=C:\Users\XPCoin\AppData\Roaming\XP\database
ErrorFile=C:\Users\XPCoin\AppData\Roaming\XP\db.log
Final lk_max_locks is 537000, sufficient for (worst case) 11 blocks in a
single transaction (up to a 5-deep reorganization)
Bound to 0.0.0.0:28192
Loading block index...
LoadBlockIndex(): hashBestChain=dd2fec3bdddbd42c7d7e  height=2136404
trust=1157501102197577422  date=03/08/18 22:16:08
LoadBlockIndex(): synchronized checkpoint
0000065a10945931644cdda436d5ad0742d085e9dee3e7fdd51b822626727e78
Verifying last 2500 blocks at level 1
 Upgrade Info: no blocktreedb upgrade detected.
 block index          235594ms
Loading wallet...
nFileVersion = 1010002
Keys: 101 plaintext, 0 encrypted, 101 w/ metadata, 101 total
 wallet                  500ms
Rescanning last 157269 blocks (from block 1979135)...
 rescan                10688ms
Loading addresses...
Loaded 10382 addresses from peers.dat  31ms
mapBlockIndex.size() = 2167392
nBestHeight = 2136404
setKeyPool.size() = 100
mapWallet.size() = 0
mapAddressBook.size() = 1
Done loading
ThreadRPCServer started
IRC seeding disabled
ThreadDNSAddressSeed started
Loading addresses from DNS seeds (could take a while)
ThreadSocketHandler started
ThreadOpenAddedConnections started
ThreadOpenConnections started
ThreadMessageHandler started
ThreadDumpAddress started
Trying to find NTP server at localhost...
ThreadNtpSamples started
Flushed 10382 addresses to peers.dat  16ms
3 addresses found from DNS seeds
```

```
ThreadDNSAddressSeed exited
trying connection 208.95.1.56:28192 lastseen=394599.5hrs
connected 208.95.1.56:28192
send version message: version 91000, blocks=2136404, us=0.0.0.0:0,
them=208.95.1.56:28192, peer=208.95.1.56:28192
socket closed
disconnecting node 208.95.1.56:28192
ThreadNtpSamples: new offset sample from 89.109.251.25, offset=2.
trying connection 131.191.98.6:28192 lastseen=17.1hrs
trying connection 195.181.244.29:28192 lastseen=394599.5hrs
connected 195.181.244.29:28192
send version message: version 91000, blocks=2136404, us=0.0.0.0:0,
them=195.181.244.29:28192, peer=195.181.244.29:28192
socket recv error 10054
disconnecting node 195.181.244.29:28192
trying connection 185.223.30.231:28192 lastseen=394599.5hrs
GetExternalIPbySTUN(18446744073709551615) returned 216.10.167.53 in attempt
2; Server=stun.zoiper.com
GetMyExternalIP() returned 216.10.167.53
AddLocal(216.10.167.53:28192,4)
```

Part 5: XP Wallet Start Up Parameters:

You can start the XP software with extra options. These are considered advanced features. This example will be over the windows wallet but should still work for mac so and Linux to a good point. But your experience may vary.

In this example we will call the wallet exe file, say where the blockchain is stored, and what XP.conf file to read.

M:\Coins\scrypt\xp\XP-qt.exe -datadir=M:\Coins\scrypt\xp\blockchain -conf=M:\Coins\scrypt\xp\blockchain\XP.conf

If we use the options -datadir and -conf we can over-ride the normal storage location, which is APPDATA by default. It also says which XP.conf file to load.

If we wanted to run a wallet as a server, daemon, and to listen for peers we would use this example:

M:\Coins\scrypt\xp\XP-qt.exe -datadir=M:\Coins\scrypt\xp\blockchain -conf=M:\Coins\scrypt\xp\blockchain\XP.conf -server -daemon -listen

Sometimes we will set application flags when we first call the wallet binary, other times we will set these flags in the XP.conf.

In addition, it also means that we can run multiple copies of xp on a machine. We need to specify another directory for blockchain and also have a copy of XP.conf in that folder. When running multiple wallets on the same server you will also have to increment the ports. Each wallet running has two (2) ports in each config.

1. Wallet 1
    a. rpcport=28191
    b. port=28192
2. Wallet 2
    a. rpcport=28193
    b. port=28194

Below we will list the full list and give a brief description of what that switch does.

```
Options:
-?                This help messages
-conf=<file>      Specify configuration file (default: XP.conf)
-pid=<file>       Specify pid file (default: XPd.pid)
-datadir=<dir>    Specify data directory
-wallet=<file>    Specify wallet file (within data directory)
-dbcache=<n>      Set database cache size in megabytes (default: 25)
-dblogsize=<n>    Set database disk log size in megabytes (default: 100)
-timeout=<n>      Specify connection timeout in milliseconds (default: 5000)
-proxy=<ip:port>  Connect through socks proxy
-socks=<n>        Select the version of socks proxy to use (4-5, default: 5)
-tor=<ip:port>         Use proxy to reach tor hidden services (default: same
as -proxy)
-torname=<host.onion>  Send the specified hidden service name when connecting
to Tor nodes (default: none)
-dns                   Allow DNS lookups for -addnode, -seednode and -connect
```

```
-port=<port>           Listen for connections on <port> (default: 28192 or
testnet: 17778)
-maxconnections=<n>    Maintain at most <n> connections to peers (default:
125)
-addnode=<ip>          Add a node to connect to and attempt to keep the
connection open
-connect=<ip>          Connect only to the specified node(s)
-seednode=<ip>         Connect to a node to retrieve peer addresses, and
disconnect
-externalip=<ip>       Specify your own public address
-onlynet=<net>         Only connect to nodes in network <net> (IPv4, IPv6 or
Onion)
-discover              Discover own IP address (default: 1 when listening and
no -externalip)
-irc                   Find peers using internet relay chat (default: 1)
-listen                Accept connections from outside (default: 1 if no -
proxy or -connect)
-bind=<addr>           Bind to given address. Use [host]:port notation for
IPv6
-dnsseed               Find peers using DNS lookup (default: 0) {1)?}
-cppolicy              Sync checkpoints policy (default: strict)
-banscore=<n>          Threshold for disconnecting misbehaving peers
(default: 100)
-bantime=<n>           Number of seconds to keep misbehaving peers from
reconnecting (default: 86400)
-maxreceivebuffer=<n>  Maximum per-connection receive buffer, <n>*1000 bytes
(default: 5000)
-maxsendbuffer=<n>     Maximum per-connection send buffer, <n>*1000 bytes
(default: 1000)
-detachdb              Detach block and address databases. Increases shutdown
time (default: 0)
-memorylog             Use in-memory logging for block index database
(default: 1)
-paytxfee=<amt>        Fee per KB to add to transactions you send
-mininput=<amt>        When creating transactions, ignore inputs with value
less than this (default: 0.00001)
-server                Accept command line and JSON-RPC commands
-testnet               Use the test network
-debug                 Output extra debugging information. Implies all other
-debug* options
-debugnet              Output extra network debugging information
-logtimestamps         Prepend debug output with timestamp
-shrinkdebugfile       Shrink debug.log file on client startup (default: 1
when no -debug)
-printtoconsole        Send trace/debug info to console instead of debug.log
file
-printtodebugger       Send trace/debug info to debugger
-rpcuser=<user>        Username for JSON-RPC connections
-rpcpassword=<pw>      Password for JSON-RPC connections
-rpcport=<port>        Listen for JSON-RPC connections on <port> (default:
28191 or testnet: 18345)
-rpcallowip=<ip>       Allow JSON-RPC connections from specified IP address
-rpcconnect=<ip>       Send commands to node running on <ip> (default:
127.0.0.1)
-blocknotify=<cmd>     Execute command when the best block changes (%s in cmd
is replaced by block hash)
```

```
-walletnotify=<cmd>    Execute command when a wallet transaction changes (%s
in cmd is replaced by TxID)
-confchange            Require a confirmations for change (default: 0)
-upgradewallet         Upgrade wallet to latest format
-keypool=<n>           Set key pool size to <n> (default: 100)
-rescan                Rescan the block chain for missing wallet transactions
-zapwallettxes         Clear list of wallet transactions (diagnostic tool;
implies -rescan)
-salvagewallet         Attempt to recover private keys from a corrupt
wallet.dat
-checkblocks=<n>       How many blocks to check at startup (default: 2500, 0
= all)
-checklevel=<n>        How thorough the block verification is (0-6, default:
1)
-par=N                 Set the number of script verification threads (1-16,
0=auto, default: 0)
-loadblock=<file>      Imports blocks from external blk000?.dat file

Block creation options:
-blockminsize=<n>      Set minimum block size in bytes (default: 0)
-blockmaxsize=<n>      Set maximum block size in bytes (default: 250000)
-blockprioritysize=<n> Set maximum size of high-priority/low-fee transactions
in bytes (default: 27000)

SSL options: (see the Bitcoin Wiki for SSL setup instructions)
-rpcssl                                  Use OpenSSL (https) for JSON-RPC
connections
-rpcsslcertificatechainfile=<file.cert>  Server certificate file (default:
server.cert)
-rpcsslprivatekeyfile=<file.pem>         Server private key (default:
server.pem)
-rpcsslciphers=<ciphers> Acceptable ciphers (default:
TLSv1+HIGH:!SSLv2:!aNULL:!eNULL:!AH:!3DES:@STRENGTH)

UI options:
-lang=<lang>           Set language, for example "de_DE" (default: system
locale)
-min                   Start minimized
-splash                Show splash screen on startup (default: 1)
```

RPC Use Tips:

Sometimes you will want to start the wallet to repair or fix transactions. These can be done by using the -zapwallettxes command at the end of the start line.

M:\Coins\scrypt\xp\XP-qt.exe -datadir=M:\Coins\scrypt\xp\blockchain -conf=M:\Coins\scrypt\xp\blockchain\XP.conf -zapwallettxes

Part 6: XP.conf: Configuring the Wallet

This file is read in by the application and applied to it on load. You will need to set the RPC Username and RPC Password to something very complex. This is very important.

If your application is not on the same server (IP Address or define a subnet) you will need to add it to the RPC Connections section.

```
rpcallowip=127.0.0.1
#rpcallowip=192.168.1.17
#rpcallowip=192.168.1.*
```

This tells the wallet to allow RPC connections from this address or network.

This is an example config file (XP.conf)

```
rpcuser=user
rpcpassword=YDFBrMHYDFE1cK2Td
# XP.conf configuration file. Lines beginning with # are comments.
# Run on the test network instead of the real XP network.
testnet=0

# Connect via a socks4 proxy
#proxy=127.0.0.1:9050
#proxy=127.0.0.1:3643

# Use as many addnode= settings as you like to connect to specific peers
addnode=seed1.xpcoin.io:28192
addnode=seed2.xpcoin.io:28192
addnode=seed3.xpcoin.io:28192
addnode=125.100.148.191
addnode=60.116.63.240
addnode=35.201.137.144
addnode=208.95.1.58
addnode=52.213.239.160
addnode=13.113.228.244
addnode=153.126.143.117
addnode=88.99.107.17
addnode=119.245.129.113
addnode=150.31.13.242
addnode=150.95.130.39

# ... or use as many connect= settings as you like to connect ONLY
# to specific peers:
#connect=188.120.246.137:7777
#connect=lv4llpu75ydlfwxgx3ej5t6dpcnyi47px4wnluf7pyxpncd5trca.b32.i2p

# Do not use Internet Relay Chat (irc.lfnet.org #XP00 channel) to
# find other peers.
noirc=1

#connect to i2p
#i2p=1

# Maximum number of inbound+outbound connections.
maxconnections=100
```

```
# JSON-RPC options (for controlling a running process)

# server=1 tells XP to accept JSON-RPC commands.
#server=0

# daemon=1 runs as daemon
#daemon=0

# tells server to listen
#listen=1

# You must set rpcuser and rpcpassword to secure the JSON-RPC api
# By default, only RPC connections from localhost are allowed.  Specify
# as many rpcallowip= settings as you like to allow connections from
# other hosts (and you may use * as a wildcard character):
rpcallowip=127.0.0.1
#rpcallowip=192.168.1.*

# Listen for RPC connections on this TCP port:
rpcport=28191
port=28192

# You can use XP or XPd to send commands to XP
# running on another host using this option:
rpcconnect=127.0.0.1

# Use Secure Sockets Layer (also known as TLS or HTTPS) to communicate
# with XP -server or XPd
#rpcssl=1

# OpenSSL settings used when rpcssl=1
#rpcsslciphers=TLSv1+HIGH:!SSLv2:!aNULL:!eNULL:!AH:!3DES:@STRENGTH
#rpcsslcertificatechainfile=server.cert
#rpcsslprivatekeyfile=server.pem

# Miscellaneous options
# Set gen=1 to attempt to generate XP
gen=0

# Pre-generate this many public/private key pairs, so wallet backups will be
valid for
# both prior transactions and several dozen future transactions.
keypool=100

# Pay an optional transaction fee every time you send XP.
#paytxfee=0.00100000

# Allow direct connections for the 'pay via IP address' feature.
#allowreceivebyip=1

# User interface options
# Start XP minimized
#min=1

# Minimize to the system tray
#minimizetotray=1
```

```
# Additionals
dns=1
checkpoints=1
dnsseed=1
banscore=5
bantime=604800
algo=scrypt
```

Part 7: Wallet RPC Calls and Console Commands:

When using the wallet with the -server -listen commands will allow the application to use the RPC system. These commands can also be run from the debug console in the application if available. Keep in mind that the wallet performance will be affected by the amount of RPC calls that are happening at one time. Avoid when possible making un-necessary wallet (RPC) calls. When possible use the function sendmany to build and send queues of transactions rather one for each transaction.

A few key commands that can really be helpful are they addnode command and the getinfo command.

Nodes or Peers:

Nodes are another word for peer. These nodes/peer wallets have both in and outbound connections. As more wallets connect and look for updated blocks, they connect to peers to get that. You can connect to 16 peers at one time and this is a wallet limit. You can add as many peers to your database as you like by using the addnode command. Buy you will only have the ability to connect to a maximum of 16 at one time.

Most users will not be peers. They will only contain outbound connections. So, they cannot service other wallets looking for blockchain data. If your using tor or i2p this may not be the case as they re-route traffic through the network bypassing any local firewall in place.

To add a node to the database, open the debug console, and type “addnode seed1.xpcoin.io add”

You can add as many nodes as you want. Just replace the address of the node for each different ip address you want to add. You can also use the (fqdn) dns name as I have used in my example.

RPC Wallet call result format:

The “getinfo” rpc call returns the status of the wallet. The result returned will be a json array.


```
{
"version" : "XP-v1.1.0.2-bdb-msvc",
"protocolversion" : 91000,
"walletversion" : 60000,
"balance" : 5.00000000,
"unspendable" : 0.00000000,
"newmint" : 0.00000000,
"stake" : 0.00000000,
"blocks" : 2137822,
"timestamping" : {
"systemclock" : 1520643578,
"adjustedtime" : 1520643579,
"ntpoffset" : 1,
"p2poffset" : 1
},
"moneysupply" : 253602170410.47509766,
"connections" : 1,
"proxy" : "",
"ip" : "216.10.167.53",
"difficulty" : {
"proof-of-work" : 0.18135152,
"proof-of-stake" : 1572.40850899
},
```

```
"testnet" : false,
"keypoololdest" : 1520468337,
"keypoolsize" : 101,
"paytxfee" : 0.00000000,
"mininput" : 0.00001000,
"errors" : ""
}
```

This data can be use in programming applications and api's. Many of the types of programs or applications developed will work through the jsonrpc process. Below is a list of wallet commands and the example syntax in use.

```
addmultisigaddress <nrequired> <'["key","key"]'> [account]
addnode <node> <add|remove|onetry>
addredeemscript <redeemScript> [account]
backupwallet <destination>
checkwallet
createmultisig <nrequired> <'["key","key"]'>
createrawtransaction '[{"txid":txid,"vout":n},...]' '{address:amount,...}'
decoderawtransaction <hex string>
decodescript <hex string>
dumpblock <hash> [destination]
dumpblockbynumber <number> [destination]
dumpprivkey <XPaddress>
dumpwallet <filename>
getaccount <XPaddress>
getaccountaddress <account>
getaddednodeinfo <dns> [node]
getaddressesbyaccount <account>
getaddrmaninfo [networkType]
getbalance [account] [minconf=1] [watchonly=0]
getbestblockhash
getblock <hash> [txinfo]
getblockbynumber <number> [txinfo]
getblockcount
getblockhash <index>
getblocktemplate [params]
getcheckpoint
getconnectioncount
getdifficulty
getinfo
getmininginfo
getnettotals
getnewaddress [account]
getpeerinfo
getrawmempool
getrawtransaction <txid> [verbose=0]
getreceivedbyaccount <account> [minconf=1]
getreceivedbyaddress <XPaddress> [minconf=1]
getsubsidy [nTarget]
gettransaction <txid>
getwork [data]
getworkex [data, coinbase]
help [command]
importaddress <address> [label] [rescan=true]
importprivkey <XPprivkey> [label] [rescan=true]
```

```
importwallet <filename>
keypoolrefill [new-size]
keypoolreset [new-size]
listaccounts [minconf=1]
listaddressgroupings
listreceivedbyaccount [minconf=1] [includeempty=false]
listreceivedbyaddress [minconf=1] [includeempty=false]
listsinceblock [blockhash] [target-confirmations]
listtransactions [account] [count=10] [from=0]
listunspent [minconf=1] [maxconf=9999999] ["address",...]
makekeypair [prefix]
mergecoins <amount> <minvalue> <outputvalue>
move <fromaccount> <toaccount> <amount> [minconf=1] [comment]
ntptime [ntpserver]
removeaddress 'address'
repairwallet
resendtx
reservebalance [<reserve> [amount]]
scaninput '{"txid":"txid", "vout":[vout1, vout2, ..., voutN],
"difficulty":difficulty, "days":days}'
sendalert <message> <privatekey> <minver> <maxver> <priority> <id>
[cancelupto]
sendfrom <fromaccount> <toXPaddress> <amount> [minconf=1] [comment] [comment-
to]
sendmany <fromaccount> '{address:amount,...}' [minconf=1] [comment]
sendrawtransaction <hex string>
sendtoaddress <XPaddress> <amount> [comment] [comment-to]
setaccount <XPaddress> <account>
settxfee <amount>
signmessage <XPaddress> <message>
signrawtransaction <hex string>
'[{"txid":txid,"vout":n,"scriptPubKey":hex,"redeemScript":hex},...]'
'[<privatekey1>,...]' [sighashtype="ALL"]
stop <detach>
submitblock <hex data> [optional-params-obj]
validateaddress <XPaddress>
verifymessage <XPaddress> <signature> <message>
walletlock
walletpassphrase <passphrase> <timeout> [mintonly]
walletpassphrasechange <oldpassphrase> <newpassphrase>
```

Part 8: Boot Strap File and Blockchain Updating:

When a user builds or installs the wallet they will be required to download the blockchain. To speed up the process we have supplied a downloadable copy of the blockchain. Download it from the following link and extract it to the locations. Make sure the wallet is off before installing the “Boot Strap”.

Link: https://drive.google.com/open?id=11mCl3gtuRNDXRAwRkC-YUon7CnjwHTGB

This will be installed to different locations depending on your OS.

1. Windows: %APPDATA%/XP/
2. Mac OS X: ~/Library/Application Support/XP/
3. Debian and Ubuntu Linux: ~/.XP/