ABGROUPCOIN

Integration

ABSTOCKS
abstocks.com

Contents

ABGroupCoin Core integration 2

What is ABGroupCoin? 2

License 2

Development Process 2

Testing 2

Automated Testing 3

Manual Quality Assurance (QA) Testing 3

Translations 3

Building ABgroupcoin 4

Unix Build Notes 4

Note 4

To Build 4

Dependencies 4

Memory Requirements 5

Linux Distribution Specific Instructions 6

Ubuntu & Debian 6

Fedora 7

Notes 7

miniupnpc 7

Berkeley DB 8

Boost 8

Security 9

Disable-wallet mode 10

Additional Configure Flags 10

Setup and Build Example: Arch Linux 10

ARM Cross-compilation 11

ABGroupCoin Core integration

https://abstocks.com/abgc

What is ABGroupCoin?

ABgroupcoin is an experimental digital currency that enables instant payments to anyone, anywhere in the world. ABgroupcoin uses peer-to-peer technology to operate with no central authority: managing transactions and issuing money are carried out collectively by the network. ABgroupcoin Core is the name of open source software which enables the use of this currency.

For more information, as well as an immediately useable, binary version of the ABgroupcoin Core software, see https://abstocks.com/abgc.

License

ABgroupcoin Core is released under the terms of the MIT license. See COPYING for more information or see https://opensource.org/licenses/MIT.

Development Process

The `master` branch is regularly built (see `doc/build-*.md` for instructions) and tested, but it is not guaranteed to be completely stable. Tags are created regularly from release branches to indicate new official, stable release versions of ABgroupcoin Core. The https://github.com/abgroupcoin/abgroupcoin-gui repository is used exclusively for the development of the GUI. Its master branch is identical in all monotree repositories. Release branches and tags do not exist, so please do not fork that repository unless it is for development reasons.

The contribution workflow is described in CONTRIBUTING.md and useful hints for developers can be found in doc/developer-notes.md.

The developer mailing list should be used to discuss complicated or controversial changes before working on a patch set.

Developer IRC can be found on Freenode at #abgroupcoin-dev.

Testing

Testing and code review is the bottleneck for development; we get more pull requests than we can review and test on short notice. Please be patient and help out by testing other people's pull requests, and remember this is a security-critical project where any mistake might cost people lots of money.

Automated Testing

Developers are strongly encouraged to write unit tests for new code, and to submit new unit tests for old code. Unit tests can be compiled and run (assuming they weren't disabled in configure) with: `make check`. Further details on running and extending unit tests can be found in /src/test/README.md.
There are also regression and integration tests, written in Python, that are run automatically on the build server. These tests can be run (if the test dependencies are installed) with: `test/functional/test_runner.py`
The Travis CI system makes sure that every pull request is built for Windows, Linux, and macOS, and that unit/sanity tests are run automatically.

Manual Quality Assurance (QA) Testing

Changes should be tested by somebody other than the developer who wrote the code. This is especially important for large or high-risk changes. It is useful to add a test plan to the pull request description if testing the changes is not straightforward.

Translations

We only accept translation fixes that are submitted through Bitcoin Core's Transifex page. Translations are converted to ABGroupCoin periodically.

Translations are periodically pulled from Transifex and merged into the git repository. See the translation process for details on how this works.

Important: We do not accept translation changes as GitHub pull requests because the next pull from Transifex would automatically overwrite them again.

Building ABgroupcoin

See doc/build-*.md for instructions on building the various elements of the ABgroupcoin Core reference implementation of ABgroupcoin.

Unix Build Notes

Some notes on how to build ABgroupcoin Core in Unix.

(For BSD specific instructions, see `build-*bsd.md` in this directory.)

Note

Always use absolute paths to configure and compile ABgroupcoin Core and the dependencies. For example, when specifying the path of the dependency:

```
../dist/configure --enable-cxx --disable-shared --with-pic --prefix=$BDB_PREFIX
```

Here BDB_PREFIX must be an absolute path - it is defined using $(pwd) which ensures the usage of the absolute path.

To Build

```
./autogen.sh
./configure
make
make install # optional
```

This will build abgroupcoin-qt as well, if the dependencies are met.

Dependencies

These dependencies are required:

Library	Purpose	Description
libboost	Utility	Library for threading, data structures, etc
libevent	Networking	OS independent asynchronous networking

Optional dependencies:

Library	Purpose	Description
miniupnpc	UPnP Support	Firewall-jumping support
libdb4.8	Berkeley DB	Wallet storage (only needed when wallet enabled)
qt	GUI	GUI toolkit (only needed when GUI enabled)
libqrencode	QR codes in GUI	Optional for generating QR codes (only needed when GUI enabled)
univalue	Utility	JSON parsing and encoding (bundled version will be used unless --with-system-univalue passed to configure)
libzmq3	ZMQ notification	Optional, allows generating ZMQ notifications (requires ZMQ version >= 4.0.0)
sqlite3	SQLite DB	Optional, wallet storage (only needed when wallet enabled)

For the versions used, see dependencies.md

Memory Requirements

C++ compilers are memory-hungry. It is recommended to have at least 1.5 GB of memory available when compiling ABgroupcoin Core. On systems with less, gcc can be tuned to conserve memory with additional CXXFLAGS:

```
./configure CXXFLAGS="--param ggc-min-expand=1 --param ggc-min-heapsize=32768"
```

Alternatively, or in addition, debugging information can be skipped for compilation. The default compile flags are `-g -O2`, and can be changed with:

```
./configure CXXFLAGS="-O2"
```

Finally, clang (often less resource hungry) can be used instead of gcc, which is used by default:

```
./configure CXX=clang++ CC=clang
```

Linux Distribution Specific Instructions

Ubuntu & Debian

Dependency Build Instructions

Build requirements:

```
sudo apt-get install build-essential libtool autotools-dev automake pkg-config bsdmainutils python3
```

Now, you can either build from self-compiled depends or install the required dependencies:

```
sudo apt-get install libevent-dev libboost-system-dev libboost-filesystem-dev libboost-test-dev libboost-thread-dev
```

BerkeleyDB is required for the wallet.

Ubuntu and Debian have their own `libdb-dev` and `libdb++-dev` packages, but these will install BerkeleyDB 5.1 or later. This will break binary wallet compatibility with the distributed executables, which are based on BerkeleyDB 4.8. If you do not care about wallet compatibility, pass `--with-incompatible-bdb` to configure.
Otherwise, you can build from self-compiled `depends` (see above).
SQLite is required for the wallet:

```
sudo apt install libsqlite3-dev
```

To build ABgroupcoin Core without wallet, see *Disable-wallet mode*

Optional (see `--with-miniupnpc` and `--enable-upnp-default`):
`sudo apt-get install libminiupnpc-dev`
ZMQ dependencies (provides ZMQ API):

```
sudo apt-get install libzmq3-dev
```

GUI dependencies:

If you want to build abgroupcoin-qt, make sure that the required packages for Qt development are installed. Qt 5 is necessary to build the GUI. To build without GUI pass `--without-gui`.
To build with Qt 5 you need the following:

```
sudo apt-get install libqt5gui5 libqt5core5a libqt5dbus5 qttools5-dev qttools5-dev-tools
```

libqrencode (optional) can be installed with:

```
sudo apt-get install libqrencode-dev
```

Once these are installed, they will be found by configure and a abgroupcoin-qt executable will be built by default.

Fedora

Dependency Build Instructions

Build requirements:

```
sudo dnf install gcc-c++ libtool make autoconf automake libevent-devel boost-devel
libdb4-devel libdb4-cxx-devel python3
```

Optional (see `--with-miniupnpc` and `--enable-upnp-default`):

```
sudo dnf install miniupnpc-devel
```

ZMQ dependencies (provides ZMQ API):

```
sudo dnf install zeromq-devel
```

To build with Qt 5 you need the following:

```
sudo dnf install qt5-qttools-devel qt5-qtbase-devel
```

libqrencode (optional) can be installed with:

```
sudo dnf install qrencode-devel
```

SQLite can be installed with:

```
sudo dnf install sqlite-devel
```

Notes

The release is built with GCC and then "strip abgroupcoind" to strip the debug symbols, which reduces the executable size by about 90%.

miniupnpc

miniupnpc may be used for UPnP port mapping. It can be downloaded from here. UPnP support is compiled in and turned off by default. See the configure options for upnp behavior desired:

```
--without-miniupnpc      No UPnP support miniupnp not required
--disable-upnp-default   (the default) UPnP support turned off by default at
runtime
--enable-upnp-default    UPnP support turned on by default at runtime
```

Berkeley DB

It is recommended to use Berkeley DB 4.8. If you have to build it yourself, you can use the installation script included in contrib/ like so:

```
./contrib/install_db4.sh `pwd`
```
from the root of the repository.

Note: You only need Berkeley DB if the wallet is enabled (see *Disable-wallet mode*).

Boost

If you need to build Boost yourself:

```
sudo su
./bootstrap.sh
./bjam install
```

Security

To help make your ABgroupcoin Core installation more secure by making certain attacks impossible to exploit even if a vulnerability is found, binaries are hardened by default. This can be disabled with:

Hardening Flags:

```
./configure --enable-hardening
./configure --disable-hardening
```

Hardening enables the following features:

- *Position Independent Executable*: Build position independent code to take advantage of Address Space Layout Randomization offered by some kernels. Attackers who can cause execution of code at an arbitrary memory location are thwarted if they don't know where anything useful is located. The stack and heap are randomly located by default, but this allows the code section to be randomly located as well.

  On an AMD64 processor where a library was not compiled with -fPIC, this will cause an error such as: "relocation R_X86_64_32 against `.....' can not be used when making a shared object;"

  To test that you have built PIE executable, install scanelf, part of paxutils, and use:

  ```
  scanelf -e ./abgroupcoin
  ```

  The output should contain:

  TYPE ET_DYN

- *Non-executable Stack*: If the stack is executable then trivial stack-based buffer overflow exploits are possible if vulnerable buffers are found. By default, ABgroupcoin Core should be built with a non-executable stack, but if one of the libraries it uses asks for an executable stack or someone makes a mistake and uses a compiler extension which requires an executable stack, it will silently build an executable without the non-executable stack protection.

  To verify that the stack is non-executable after compiling use: `scanelf -e ./abgroupcoin`

  The output should contain: STK/REL/PTL RW- R-- RW-

  The STK RW- means that the stack is readable and writeable but not executable.

Disable-wallet mode

When the intention is to run only a P2P node without a wallet, ABgroupcoin Core may be compiled in disable-wallet mode with:

```
./configure --disable-wallet
```

In this case there is no dependency on Berkeley DB 4.8 and SQLite.

Mining is also possible in disable-wallet mode using the `getblocktemplate` RPC call.

Additional Configure Flags

A list of additional configure flags can be displayed with:

```
./configure --help
```

Setup and Build Example: Arch Linux

This example lists the steps necessary to setup and build a command line only, non-wallet distribution of the latest changes on Arch Linux:

```
pacman -S git base-devel boost libevent python
git clone https://github.com/abgroupcoin/abgroupcoin.git
cd abgroupcoin/
./autogen.sh
./configure --disable-wallet --without-gui --without-miniupnpc
make check
```

Note: Enabling wallet support requires either compiling against a Berkeley DB newer than 4.8 (package `db`) using `--with-incompatible-bdb`, or building and depending on a local version of Berkeley DB 4.8. The readily available Arch Linux packages are currently built using `--with-incompatible-bdb` according to the PKGBUILD. As mentioned above, when maintaining portability of the wallet between the standard ABgroupcoin Core distributions and independently built node software is desired, Berkeley DB 4.8 must be used.

ARM Cross-compilation

These steps can be performed on, for example, an Ubuntu VM. The depends system will also work on other Linux distributions, however the commands for installing the toolchain will be different.

Make sure you install the build requirements mentioned above. Then, install the toolchain and curl:

```
sudo apt-get install g++-arm-linux-gnueabihf curl
```

To build executables for ARM:

```
cd depends
make HOST=arm-linux-gnueabihf NO_QT=1
cd ..
./autogen.sh
./configure --prefix=$PWD/depends/arm-linux-gnueabihf --enable-glibc-back-compat --enable-reduce-exports LDFLAGS=-static-libstdc++
make
```

For further documentation on the depends system see README.md in the depends directory.