Installing and configuring qOp on 32-bit Windows

Version 2.5.0.1 (October 2017)

This document describes how to install, on the Windows 32-bit operating system, the qOp package and certain software packages that are pre-requisites for it. If you do not have the MinGW environment for C/C++ programs installed, please see the appendices.

This document uses the following typeface conventions:

- Commands for you to enter are shown in a fixed-width font, such as: `ps -ef`
- File and directory names are shown in a bold fixed-width font, such as: **`foo.bar`**
- The shell prompt is indicated via `$` and so the `cp` command would appear as: `$ cp`

1. Installing qOp from the installation file

The qOp installation file should have been provided to you. The installation file is named:

`qOp.win32_2.5.tar.gz`

A. Decide whether to do a single- or multi-user installation

The qOp package may be installed for a single user (typically on a laptop or desktop system) or for multiple users (typically on a shared server). For a single user, it is often simplest to install it in your MinGW home directory, which by default is `C:\MinGW\msys\1.0\home\%USERNAME%`. For multiple users, `C:\Program Files (x86)\` is a common choice of install directory. This document assumes that you choose a directory to which we append "`\qOp`" and refer to it as the installation directory or `DWAVE_HOME` (later in this document that value will be stored in the environment variable `DWAVE_HOME`).

B. Move the qOp installation file into the installation directory

Move the qOp installation file into the installation directory.

C. Confirm the contents of the installation file

If you have reason to believe the download or transfer of the qOp tar-file may have introduced errors, you can confirm that it was downloaded correctly with the following steps.

Start a Unix shell window (via MinGW) and examine the installation file as follows:

```
$ shasum -c qOp.win32_2.5.tarGzShasum.txt
```

- Response should be
  `> qOp.win32_2.5.tar.gz: OK`
- If not an OK response, there was a problem with the download

```
$ gunzip qOp.win32_2.5.tar.gz
```

```
$ shasum -c qOp.win32_2.5.tar.shasum
```

- Response should be
  `> qOp.win32_2.5.tar: OK`
- If not an OK response, there was a problem with the unzip

Note that the gunzip invocation above is redundant with the gunzip in the next section.

D. Open the qOp installation file and install the software

Start a Unix shell window and do the following:

```
$ gunzip qOp.win32_2.5.tar.gz
```

This should return to your shell prompt, with no error. This step converts the file from a "GZIPped TAR file" into a "TAR file." (The original file was compressed so that it required less space. A TAR file is an archive file, allowing storing files in a "directory" structure.")

```
$ tar xvf qOp.win32_2.5.tar
```

The screen will print several lines of text quickly. This "untars" the "TAR file", creating the desired directory structure, and installs the qOp software. This will create a directory named qOp in the installation directory.

E. Inspect the qOp software structure

In your MinGW Unix-like shell window, type

```
$ cd qOp
```

which means "change directory", and moves you into the qOp directory. Then, do

```
$ ls -als
```

which is the Unix command `ls`, for "list", which shows the files in the directory. The "-als" options cause the list to include "dot files" (those starting with "`.`", which are normally omitted by `ls`) and makes the output more readable.

Here are the top-level contents of the qOp directory:

Name	Type	Contents
`.dwrc`	Initialization file	Connection information for the local simulator
`Quantum Apprentice.xlsm`	Excel Spreadsheet	Visualization and simulation tool
`bin`	Directory	Executables: `toq.exe`, `dw-exec.exe`, `qbsolv.exe`, `qsage.exe`, etc.
`doc`	Directory	Documents for C API, man pages for utilities
`dwave_qbsolv`	Python package	Enables Python calls to qbsolv
`dwave_sapi.h`	C header file	

`dwave_sapi.dll`
`dwave_sapi.lib`
`libgnurx-0.dll`
	C libraries	SAPI 2.6 library

`epqmi.h`	C header file	For use by programs calling functions in libepqmi.a
`examples`	Directory	Example source code for C API, ToQ, dw, qsage, and qbsolv/QUBOs
`libepqmi.a`	C dynamic library	Library version of dw functionality

licenses	Directory	Licenses
qOp-version.txt	Text file	Version information
qOp_2.5_Release_Note.txt	Text file	Information about the qOp 2.5 release
qOp_win32_INSTALL.pdf	PDF file	This document
src	Directory	Source code, namely of qbsolv, for building dwave_qbsolv
workspace.c4 workspace.dw2x_vfyc workspace.dw2000q_vfyc	Workspace directories	QMI and SOL files for the 128-qubit simulator, and the 1152-qubit and 2048-qubit virtual full yield solvers

2. Configuring the system for qOp

A. Edit your ~/.bash_profile and ~/.bashrc files

Next you will need to edit your **~/.bash_profile** and **~/.bashrc** files to include configuration information for the qOp package. In your MinGW Unix-like shell session, change to your home directory via:

```
$ cd
```

Append the following lines to your ~/**.bashrc** file, using **vi** or any text editor you prefer:

```
export DWAVE_HOME=$HOME/qOp                # see note below
export DWAVE_HOME=<multiuserInstallDirectory>/qOp
export DWAVE_WORKSPACE=$HOME/qOp           # see note below
export PATH=$PATH:$DWAVE_HOME:$DWAVE_HOME/bin
export BASH_ENV=$DWAVE_HOME/bin/dw_setup   # see note below
. $BASH_ENV
```

Note: You should include only one of the two green lines that set the DWAVE_HOME environment variable. If you're installing qOp as a single-user installation in your home directory, use the first line. If qOp was installed for multiple users, you will need to find the name of the directory where it was installed (i.e., replace "<multiuserInstallDirectory>" with the actual directory name).

Note: If you are running a multi-user qOp installation, setting DWAVE_WORKSPACE is the way to control where in your file structure qOp will store temporary and configuration files. If you are running a single-user qOp installation, you may omit setting DWAVE_WORKSPACE.

Note: The name of the **dw** set-up script changed between qOp 2.3.1 and 2.4. To avoid confusion with the **dw** shell command, the set-up script is now called **dw_setup**. If you are upgrading from qOp 2.3.1 or an earlier release, you will need to change the line above in your **~/.bashrc** file.

Now add the lines below to your ~/**.bash_profile** to assure it calls the ~/**.bashrc** file. It is not necessary to have this in your ~/.bash_profile more than once:

```
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi
```

Exit your shell and start a new session to test that your configuration is able to correctly locate the executables by issuing these `which` commands in your shell:

```
$ which qbsolv -V
```

```
$ which dw-exec
```

If you see an "unknown command" or empty response to either of these commands, check your `~/.bashrc` file and make sure that the location specified for the qOp executables is correct before proceeding.

After completing these configuration steps, you should be able to start a new shell and invoke `qbsolv` as follows:

```
$ qbsolv
```

You should see output like the following:

```
 Version open source 2.5
 Compiled: Sep 18 2017,13:36:05
```

that indicates qbsolv has executed properly.

Also try

```
$ dw
```

You should see output that looks like this:

```
USAGE: dw {get|set|cd|pwd|embed|bind|exec|val|add|trans|ls|mv|cp|touch|rm|...
```

B. Change the Windows PATH so that the qOp components will run

The executables included with qOp are dynamically linked and so you need to inform Windows where to look in to find the relevant dynamically linked libraries (DLLs). Do this by setting the Windows `PATH` environment variable in the Windows Control Panel. (Note: both Windows and MinGW have environment variables; creating these environment variables in the right context is essential.) The specific interface used to access environment variables depends on the version of Windows that you are using. Edit the setting for the `PATH` environment variable. If you are running a single-user qOp installation, add this component to the `PATH` environment variable:

```
C:\MinGW\msys\1.0\home\%USERNAME%\qOp
```

Replace `%USERNAME%` with the actual name of the directory where your shell starts. If you are running a multi-user qOp installation, add the folder where qOp was installed, typically:

```
C:\Program Files (x86)\qOp
```

Note that either of these is a Windows path name and it uses backslashes to separate the components of the directory, and it begins with `C:` to specify the drive letter. The edits you made to the `~/.profile` file use Unix syntax, which requires forward slashes to separate components of the directory and maps the `C:` drive letter into a directory name.

After completing these configuration steps, you should be able to start a new shell and invoke `toq` as follows:

```
$ qbsolv
```

You should see output like the following:

```
 Version open source 2.5
 Compiled: Sep 18 2017,13:36:05
```

that indicates qbsolv has executed properly.

Similarly, you may want to confirm that the dw command executes properly. At the shell prompt, typing

```
$ dw
```

should evoke a usage message something like

```
USAGE: dw {get|set|cd|pwd|embed|bind|exec|val|add|ls|mv|cp|touch|rm|…
```

It is possible that you are missing a Microsoft package called "Microsoft Visual C++ 2013 Redistributable (x86)." When you type "`qbsolv`", if a particular DLL file `msvcp120.dll` is missing, then the free Microsoft DLL needs to be obtained.

C. Change the ~/.dwrc file to connect to your D-Wave system

In the `$DWAVE_HOME` directory, you will see a file named `.dwrc`. This contains configuration information that enables you to quickly point your program for any of the qOp tools to your D-Wave hardware system or to a D-Wave software simulator included in the D-Wave software. Copy that file to your home directory, for example via

```
$ cd
$ cp $DWAVE_HOME/.dwrc .
```

The released `.dwrc` file contains a single entry consisting of

```
laptop|local
```

where `laptop` is the name you would use to refer to it and `local` is a distinguished name indicating to use the simulator running on the system where qOp is installed. Running in the simulator is the easiest way to get started, as it eliminates some possible sources of failure.

At some point you will want to run on your D-Wave hardware system, so you will need to add an entry pointing to that. The format of such an entry is "`name|sapiURL,token`" or "`name|sapiURL,token,proxyURL`" if your D-Wave system is accessed via a proxy server. You'll probably want to contact the system administrator for your D-Wave system to get the SAPI URL and optionally proxy URL. Your token needs to be created via the Qubist interface. You will also want to learn the name(s) of the solvers on that system to which you have access. An example entry in `~/.dwrc` might be

```
mydwave|https://qubist.myorg.com/sapi,87e154d397123c40db123c
```

Once you've added this to your `~/.dwrc` file, you can direct execution of qOp programs to the simulator via

```
$ dw set connection laptop
$ dw set solver c4-sw_sample
$ qbsolv -i program.qubo
```

or to your D-Wave hardware system named `mydwave` via

```
$ dw set connection mydwave
$ dw set solver <mysolvername>
$ qbsolv -i program.qubo
```

At this point you are ready to create programs in any of the qOp tool input formats and execute them locally on the simulator or on hardware.

Note: The qOp 2.5 release includes pre-built workspaces for virtual-full-yield (VFY) solvers and depends on a new manual process for creating a pre-built workspace for a given D-Wave system. If you run on a D-Wave system and not via VFY solvers, please contact your D-Wave system administrator for access to the workspace specific to your D-Wave system. Similarly, if there is an error in setting up these pre-built workspaces, qbsolv may put out a message "No pre-embedding found", in which case please contact your D-Wave system administrator.

D. Install the Python wrapper for qbsolv

As of qOp 2.5, qbsolv has a Python wrapper (`dwave_qbsolv`) that is callable as a Python function. The simplest way of installing `dwave_qbsolv` is the following, which can be executed independent of the current directory:

```
$ pip install dwave_qbsolv
```

Some platforms or releases may not have pre-built binaries, in which case the method above will not work and building from source is required, which can be done by the following:

```
$ cd $DWAVE_HOME/src/qbsolv/python
$ pip install -r requirements.txt
$ python setup.py install
```

Whichever method you use for installing `dwave_qbsolv`, to confirm it is properly installed, you can execute the `tryDwaveQbsolv.py` example from the `examples/qbsolv` directory.

E. Run the provided example programs

In the `$DWAVE_HOME/examples` directory are several examples intended to be useful, and scripts that build or execute them. All of the scripts whose names end in `.bash` are intended to be executed with the bash shell.

Use the qOp-examples command (alternatively, **`dw examples`**) to compile and run all the examples. The examples illustrate use of qbsolv, dw, ToQ, qsage, and C/SAPI. Run them as follows:

```
$ qOp-examples
```

This command will copy the contents of $DWAVE_HOME/examples to a new directory within your current working directory. After copying the files, the command will compile the various examples as necessary. You should see a single line of output for each C program compiled by the shell script, *e.g.*:

```
*************** C: Compiling sapi_connectingToSolver ***************
```

Additionally, you should see a few lines for each QUBO embedded into the simulator:

```
*************** dw: Embedding 1of3 ***************
```

After preparing all examples to run, the command will execute each example. Standard output from each of these jobs is captured in a file whose name matches the stem of the input file and whose suffix is .out.

The `$DWAVE_HOME/doc` directory contains "man" pages for the ToQ, Quantum Apprentice, `dw`, and `qsage` tools, as well as the `.qubo` and `.q` file formats. (The user guides for the C Solver API are also in that directory.)

F. If needed, create pre-embedding files specific to your system

The `qbsolv` tool uses a pre-embedded file to save time when executing. If you anticipate running on solvers that are specific to the D-Wave system you use, you may need to create embedding(s) that are specific to the system you use. If you only use virtual-full-yield (VFY) solvers, you can skip this step. If, however, you use solvers that are exactly configured to the hardware of the system you use, you need to run the `makefull.sh` script that resides in the `bin` directory in the qOp release. Start with the file `bin/ReadMemakefull.txt` for instructions.

G. Copy `toq` configuration files into all directories where `toq` programs will be executed

The simplest way to configure the qOp tools to a particular execution target, such as the software simulator or your specific D-Wave system, is via `dw set`, as described just above. If you use that approach, you can skip this section.

If you do not use the `dw set` approach, there are several files in your `$DWAVE_HOME/examples` directory that make it more convenient to run `toq`. If you only run `toq` in the `examples` directory, you need not do anything, but if you want to run in some other directory (say `$HOME/mytoq`), there is one file you must copy, and several others that may make it easier to run `toq` programs. The file you will need is the `toq` startup file, `.toqrc`, which sets the `toq` configuration by default to run on the simulator. To run `toq` in `$HOME/mytoq`, enter the following commands:

```
$ cd $HOME/examples
$ cp -p .toqrc $HOME/mytoq
$ cd $HOME/mytoq
$ ./toq -r -i mypgm.toq
```

This final command will execute `toq` on your program `mypgm.toq`.

You can get the other configuration files into `$HOME/mytoq` by executing the following commands (note that these files provide shortcuts, but are not required to execute `toq`).

```
$ cd $DWAVE_HOME/examples
$ cp -p toqREADME confSim confHdw15 confHdw22 toqSim toqHdw  toqHdw15
        toqHdw22  toqSim2  $HOME/mytoq
```

(Note that this last command is all on one line.) Review the file `toqREADME` for background on how to use these configuration files.

Appendix A: Contents

qOp contains the following components:

1. Quantum Apprentice: an Excel spreadsheet, which allows the user to manipulate, visualize and execute problems on the simulator and also quantum hardware.
2. `toq`: compiles and executes problems written in a prototype language-input format which defines a constraint satisfaction problem.
3. `qsage`: optimizes a user-provided C-language objective function.
4. `qbsolv`: executes quadratic unconstrained binary optimization (QUBO) programs, even if they happen to be bigger than will fit in the D-Wave hardware or simulator being used.
5. dw: provides shell access to many common API functions such as creating a connection, obtaining a solver, initializing and executing a QMI, and examining samples.
6. Solver API (SAPI) C library: invokes the low-level D-Wave interface routines. It contains a standard header file and accompanying library file. You can use it to compile and link C programs that directly use the low-level API. Example code is provided which shows how to do this. A separate document describes this API in detail.

Before installing qOp, you may need to install and/or configure certain software packages upon which qOp depends.

Appendix B: Prerequisites for C/C++-language programs

Before installing qOp, you will need to install and configure the MinGW package. This will allow you to compile and link programs using the D-Wave C language API, which is used internally by the qOp tools. It will also allow you to open Linux shell windows, and run Linux shell commands, on a Windows machine.

A. MinGW

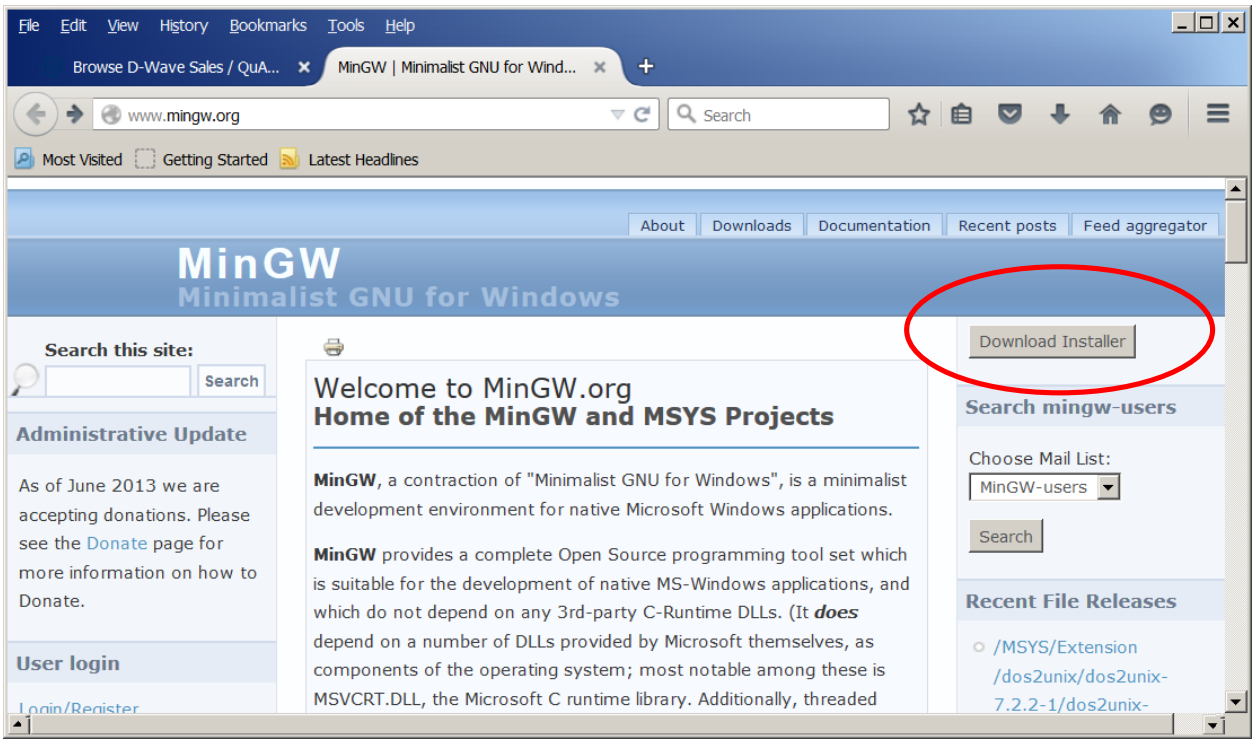
You can read about MinGW here:

http://www.mingw.org/

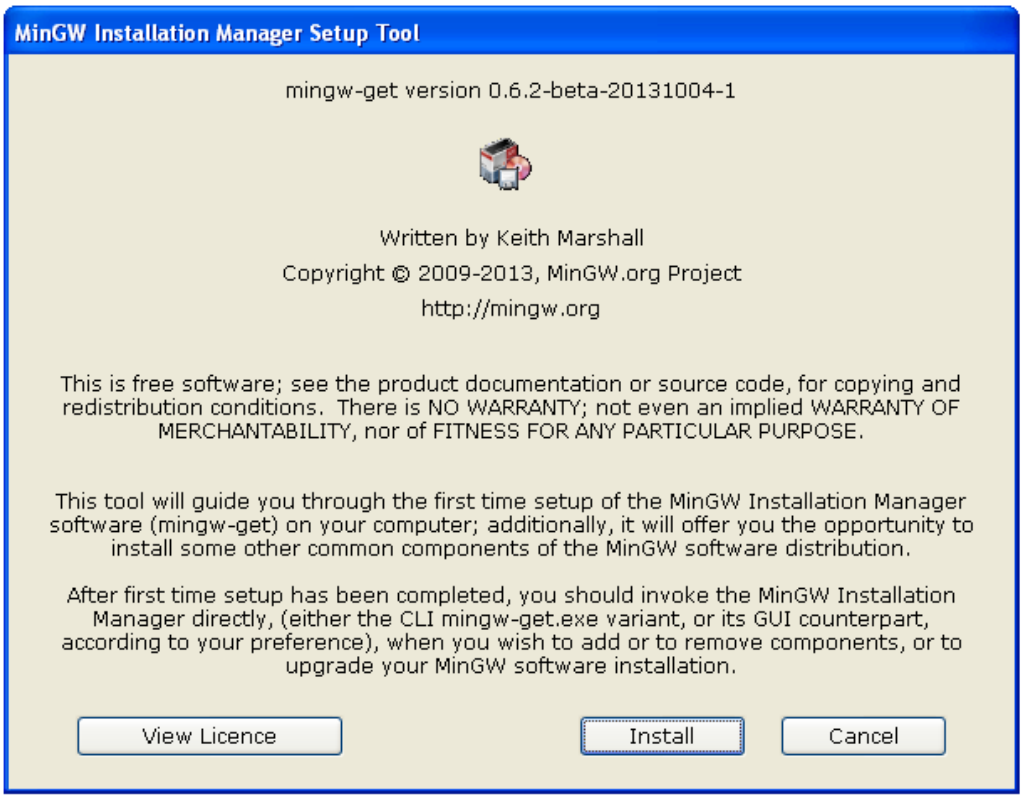
or here:

http://sourceforge.net/projects/mingw/

MinGW provides a minimalist GNU environment for Windows. Download the installer for MinGW using the Download link on either of the two links just provided:



The downloaded utility is the MinGW Installation Manager Setup Tool. Please save it to your Desktop, and then double-click it. It starts something like this:



Click on Install. On the first screen, just leave the default location for installation, which is:

```
C:\MinGW
```

Leave the click boxes alone, and click Continue. MinGW will update its own software catalog – you will see some green progress boxes – and then click Continue when it is finished. (It will say something like, "Updated 111 of 111 catalogs").

Now, you will see a "MinGW Installation Manager" window, with some check boxes on the left side. Select the following packages for installation:

1. mingw-developer-toolkit
2. mingw32-base
3. mingw32-gcc-g++

When you try to click each one of these three boxes, you may have to right click also, on "Mark for Installation." You should see little arrows in the three boxes, and there will also be an arrow in the box for "msys-base."

Now, go to the Installation tab on the MinGW Installation Manager window, and pull it down to "Apply Changes." A box will ask you if you want to proceed, and click Apply. This will begin the installation of the packages. The installation process may continue for a while – there are many sub-packages to install.

When the installation is finished, the screen should say, "All changes were applied successfully – you may close this dialogue." Click Close. Now, in the MinGW Installation Manager window, you should see the clicked boxes showing up in green, and the "Installed Version" column should have data in it for those four lines. You can then close the MinGW window by clicking the red X in the upper right.

B. Creating a MinGW shortcut

After installing the above three MinGW packages, create a short-cut on your desktop which will start the MinGW shell. Right-click on your desktop and select `New` → `Shortcut`. The location of the item for which the shortcut will be created is:

```
C:\MinGW\msys\1.0\msys.bat
```

Name this shortcut `MinGW Bash`. When you double-click this shortcut, a new Bash shell will start. This is a terminal session that emulates standard Unix shells. Many standard Unix commands are available in the terminal session that will be created by using the shortcut.

C. Create your first Unix shell window

Double click the `MinGW Bash` icon that you just created, and a shell window should show up, with something that looks like this:

```
cztester@WIN ~

$
```

The $ is the "Unix shell prompt" that we will be using going forward. When we mention "type something at the Unix shell prompt," or "at the shell prompt," we mean to begin typing after the dollar sign.

D. Checking if the C and C++ compilers work

Open a Unix shell window (by double clicking on `MinGW Bash`, as discussed in the previous section). At the dollar sign, type

```
$ which gcc
```

```
$ which g++
```

If the output from either of these commands is “unknown command” or a blank response, then you will need to repair your MinGW install.

The current version of these compilers in MinGW is 4.9.3, but it may have depended on the Windows version that you are running. For our purposes, the version number doesn’t matter.