How to Build A BViewer Installation File

by Tom Atwood **January 31, 2019**

The Development Environment

The BViewer software "projects" are maintained and built using the Microsoft Visual Studio integrated development environment (IDE). As of this writing, the Community Edition of Visual Studio (VS) 2017 is the current version. The VS Community Edition software is available free from Microsoft.

The BViewer projects are written in the C++ programming language. Include files that define symbols and data structures use the .h extension. Source files containing data declarations and executable functions use the .cpp extension. Occasionally, files written in the C programming language are included as well. These files have the .c extension.

The BViewer Project Organization

The three primary BViewer software projects are:

- BViewer: This project builds BViewer.exe, the BViewer software which displays the GUI and supports Dicom image interpretation. BViewer.exe displays the study image, displays standard reference images for comparison, acquires the B-reader data input comprising the interpretation, and displays the finalized report form. You can build this project by starting Visual Studio 2017, selecting the menu item File - Open - Project/Solution and specifying the file \BViewer\BViewer.sln as the solution file.
- BRetriever: This project builds BRetriever.exe, which runs as a Windows service. As such, it runs in the background and does not have a GUI. For debugging purposes it can be run as a console application by setting a flag in ServiceMain.cpp and rebuilding. Care should be taken that this flag is set as

  ```
  BOOL                                    bRunAsService = TRUE;
  ```

  when building a release version. You can build this project by starting Visual Studio 2017, selecting the menu item File - Open - Project/Solution and specifying the file \BRetriever\BRetriever.sln as the solution file.

- ServiceController: The ServiceController application is activated by clicking the *Control BRetriever* button on the *Set Up BViewer* tab. This displays a small GUI that supports installing, uninstalling, starting and stopping the BRetriever service. This project is usually current and does not often need to be rebuilt. You can build this project by starting Visual Studio 2017, selecting the menu item File - Open - Project/Solution and specifying the file \ServiceController\ServiceController.sln as the solution file. (NOTE: This service can also be started, stopped, etc., using the general-purpose Windows service control application included in the Windows Control Panel.)

In addition to these three software projects, there are several software libraries that are included during the project linking. These libraries rarely need to be rebuilt, but this has happened in the past when Microsoft made changes in their development environment that made certain function calls obsolete or when library functionality needed to be upgraded.

If the libraries need rebuilding, the resulting .lib files need to be copied into the \BViewer\lib and the \BRetriever\lib folders as appropriate, where they can be found by the linker for building these two projects. There are separate versions of each library for release and for debugging.   (Debug versions include source code symbols for the debugger and their file name usually ends in "d".)

(ASIDE: When BViewer was originally created, JPEG image files typically contained images where each pixel is represented by a single 8-bit byte. For grayscale images this allows 256 different shades of gray, going from white to black (or vice-versa). Colored images represent each pixel with three separate RGB "component" bytes grouped together. If degrees of transparency are involved there is a fourth byte to specify this for each pixel. This image pixel structure is what Microsoft Windows is based on. At the time, Microsoft did not support pixel grayscale resolutions higher than 8 bits, so special graphics interface software was required to achieve the 10-bit resolution required for digital chest radiograph image interpretation. Radiographic images are frequently embedded within Dicom image files as JPEG images with 8, 12 or 16 bits per pixel. To be able to read and display them, BViewer uses any one of the three JPEG libraries, depending upon the image specifications, to encode or decode these images.)

The associated library projects are

- Jpeg8: This supports 8-bit JPEG image file encoding and decoding. JPEG2000 is not supported by this open-source project. You can build this project by starting Visual Studio 2017, selecting the menu item File - Open - Project/Solution and specifying the file \Jpeg8\Jpeg8.sln as the solution file. This library is written in the C programming language.

- Jpeg12: This is a modified version of the Jpeg8 library which supports 12-bit JPEG image file encoding and decoding. JPEG2000 is not supported by this open-source project. You can build this project by starting Visual Studio 2017, selecting the menu item File - Open - Project/Solution and specifying the file \Jpeg12\Jpeg12.sln as the solution file. This library is written in the C programming language.

- Jpeg16: This is a modified version of the Jpeg8 library which supports 16-bit JPEG image file encoding and decoding. JPEG2000 is not supported by this open-source project. You can build this project by starting Visual Studio 2017, selecting the menu item File - Open - Project/Solution and specifying the file \Jpeg16\Jpeg16.sln as the solution file. This library is written in the C programming language.

- libpng: This supports PNG image file encoding and decoding. This project builds both a PNG library and a ZLIB library. You can build this project by starting Visual Studio 2017, selecting the menu item File - Open - Project/Solution and specifying the file \libpng\lpng1234\lpng1234\projects\visualc71\libpng.sln as the solution file. This library is written in the C programming language.

If changes are made in certain include (.h) files in these library projects, these include files need to be updated (copied) into the BViewer and BRetriever project folders.

Project Versions

Each software project is configured for building either a "Release" version or a "Debug" version. These build the 32-bit versions of the projects.

You may also notice 64-bit configuration options, which *you should ignore*. These were abandoned after it was found that Microsoft had modified their sockets interface in the 64-bit version. This would have required re-engineering these interfaces, which was deemed at the time to be not worth the effort.

Help File Generation

There are two help file projects, one for the NIOSH mode of BViewer operation and one for the General Purpose (GP) mode of operation. These have been built with the extremely antiquated Microsoft HTML Help Workshop software (hhw.exe) which is no longer supported, but is still available on the internet.

The BViewer.chm help file corresponds to the NIOSH mode of BViewer operation. The BViewerGP.chm help file corresponds to the GP mode of operation. Both files are included in each distribution, and BViewer selects the appropriate one at runtime depending upon its mode of operation. (The BViewer mode of operation is set within the BViewer.cfg file and is established at program startup.)

Within the BViewer project, the \hlp folder is used to generate the BViewer.chm file and the \GPhlp folder is used to generate the BViewerGP.chm file. If updated, these files must be copied into the \Install\Files folder before generating a new installation file.

Creating the Installation File

Creating an installable file is handled by the Inno Setup software, which is available as freeware on the internet. The installation files and specifications are included in the \Install subdirectory of the BViewer project.

In order to produce an updated installation file, first make any development changes in the source code and build the release versions of the associated executables.

Copy the newly modified executable(s) and/or any other modified files into the \Install\Files directory. The BViewer_1_0.iss file tells the Inno setup software where to find each of these files, and where to put each file during an installation. After the modified files have been copied, run the Inno setup software on the BViewer_1_0.iss file. From the “Build” item on the main menu, select the “Compile” option. This will cause the installation to be rebuilt. This produces a new “Setup.exe” file in the \Install\Output directory. This file then needs to be renamed to reflect its software version and date of generation, for example, *Setup_2g_12172018.exe*.

The following is a checklist for generating an installation file:

1. Change the BViewer version numbers in:

   BViewer.rc
   Module.cpp
   AboutBViewer.txt
   Mainfrm.cpp

2. Change the version numbers in the BRetriever file ServiceMain.cpp.
3. Change the version number in ServiceController if it is updated.
4. Rebuild the release versions completely.
5. Move the rebuilt .exe files from their project \Release folders to \BViewer\Install\Files.
6. Run the Inno Setup program, opening the BViewer script file, BViewer_1_0.iss. Update the version number in this file.
7. In the Inno Setup program, select the Build - Compile menu item. This causes the new installation file to be created in the \BViewer\Install\Output folder.
8. Rename the file to designate the version number and date of the enclosed software.