Building Libraries and Executables Supporting RiverWare

**By Neil Wilson**

**CADSWES**

The purpose of this document is to pull together information for building the libraries that are linked into RiverWare along with supporting executables.

# RiverWare

RiverWare requires a number of third-party and CADSWES-created libraries. Sections below describe each of these .

## Qt

Qt is the GUI library for RiverWare, and also provides a number of other capabilities (Container classes, networking, XML, etc.). Following are steps for building Qt showing version 4.8.4 as an example:

(1) Download file qt-everywhere-commercial-src-4.8.4.zip from the [Qt4 commercial customer portal](http://qt.digia.com/Log-in-Customer-Portal/) (http://qt.digia.com/Log-in-Customer-Portal/).

* Also, since we are using the commercial version, the Qt licence file is needed to run the "configuration" program. If that file isn't in your home directory (e.g. C:\Users\philw\.qt-license) then also download that from the customer portal and copy it to your home directory.

(2) Extract and move the top-level directory to C:\Riverware\tools\Qt-484\

(3) Make a backup copy of moc.prf:

* copy: C:\Riverware\tools\Qt-484\mkspecs\features\moc.prf
* to: C:\Riverware\tools\Qt-484\mkspecs\features\moc\_prf\_ORIG.txt
* to: C:\Riverware\tools\Qt-484\mkspecs\features\moc\_prf\_NEW.txt

Edit the "NEW" file to change the following line and copy it to replace moc.prf:

* original: win32:count($$list($$INCLUDEPATH), 40, >) {
* change to: win32:count($$list($$INCLUDEPATH), 80, >) {

The purpose is to change the number of lines that can be present in the include path before a temporary file (mocinclude.tmp) is created to contain the path. This file, when created, is included in the dependencies for the project and when recreated with each build was causing the project to always be out of date and require rebuilding. Increasing the number allows our Qt…Generated projects to not require the temporary file and not always be out of date with every build.

(4a) Put C:\Riverware\tools\Qt-484\bin on the path. (This should be removed before building RiverWare).  
(4b) Remove C:\RiverWare\tools\UnxUtils from the path. (This must be restored before building RiverWare).

(5) Open one of the two Visual Studio 2010 command prompts, depending on whether a 32-bit or 64-bit build of Qt is desired. From the Windows Start Menu: All Programs >> Microsoft Visual Studio 2010 >> Visual Studio Tools >> ...

* *32-bit Build:*  Visual Studio Command Prompt (2010)
* *64-bit Build:*  Visual Studio x64 Win64 Command Prompt (2010)

... confirm that the path changes worked. (type: path). Note that the path will start with a whole bunch of Visual Studio paths. The configured paths follow those.

(6a) Change to the C:\Riverware\tools\Qt-484\ directory ...

* C:
* cd C:\Riverware\tools\Qt-484\

(6b) Copy and paste the following 'configure' command (*all on one line*) ...

* configure.exe -debug-and-release -commercial -platform win32-msvc2010 -shared -no-exceptions -qt-zlib -qt-libpng -qt-libmng -qt-libjpeg -no-libtiff -webkit

... the first part of this (the Qt config process) takes about 10 or 15 minutes to run on our 64-bit, Windows 7 machines and 2 or 3 times that on my 32-bit Windows XP machine. It is then necessary to respond to the following prompt which starts the build of the Qt libraries, tools and examples. This takes about four to five hours on 64-bit, and maybe 12 hours on 32-bit. (Just type "nmake") ...

Qt is now configured for building. Just run **nmake**.  
To reconfigure, run nmake confclean and configure.

(7) Qt 4.8.4 DLLs are generated in this directory: C:\Riverware\tools\Qt-484\lib\.  There are 19 DLLs in both debug and non-debug (release) versions (38 in all). Copy the 38 generated DLLs:

* From:  C:\Riverware\tools\Qt-484\lib\
* To:  C:\Riverware\tools\bin\

(8) Restore the system path. (See step 5 above). This must be done before building RiverWare. In particular, the Unix Utilities are needed for the RiverWare build.

## Qwt

Qwt is the plotting library used for RiverWare plots. Following are steps for building Qwt using version 5.2.2 as an example.

(1) Download the Qwt 5.2.2 zip file from <http://sourceforge.net/projects/qwt/files/qwt/5.2.2/>

* qwt-5.2.2.zip (2011-08-01; 2.9 MB)

(2) Unpack and move the top level directory to:

* C:\Riverware\tools\Qwt-522\_Qt-484\

(3) Look over the INSTALL file instructions, in particular, Section (B) Win32/MSVC Qt3/Qt4

(2) Edit qwtconfig.pri (a 'pro' include file) -- significant values:

INSTALLBASE = C:/RiverWare/tools/Qwt-522\_Qt-484  
CONFIG += debug\_and\_release  
CONFIG += build\_all  
# CONFIG += QwtDll  
CONFIG += QwtPlot  
CONFIG += QwtWidgets  
# CONFIG += QwtSVGItem  
# CONFIG += QwtMathML

(4) Make a backup copy of qwt\_plot\_curve.cpp:

* copy:  C:\Riverware\tools\Qwt-522\_Qt-484\src\qwt\_plot\_curve.cpp
* to:  C:\Riverware\tools\Qwt-522\_Qt-484\src\qwt\_plot\_curve-ORIG.cpp
* to:  C:\Riverware\tools\Qwt-522\_Qt-484\src\qwt\_plot\_curve-NEW.cpp

(5) Apply SOURCE CODE FIX to qwt\_plot\_curve-NEW.cpp and copy over qwt\_plot\_curve.cpp.  
See the edited copy [posted here](file:///R:\doc\Qt\2012\Qt483\Qwt522-Changes\qwt_plot_curve-NEW_cpp.txt):  
   http://cadswes2.colorado.edu/~philw/2012/Qt483/Qwt522-Changes/qwt\_plot\_curve-NEW\_cpp.txt

*Method:* void QwtPlotCurve::drawLines (QPainter \*painter, const QwtScaleMap &xMap, const QwtScaleMap &yMap, int from, int to) const *... around line 740*

|  |
| --- |
| for (int i = from; i <= to; i++) {  // CADSWES FIX -- originally developed with Qwt 5.0.2:  // Change [i] to [i - from]. Without this, our solution to   // drawing disjoint curves (having internal NaNs) crashes.  //   //-- WRONG: QwtDoublePoint &p = points[i];  QwtDoublePoint &p = points[i - from];    p.setX( xMap.xTransform(x(i)) );  p.setY( yMap.xTransform(y(i)) ); } |

(5) Optional: Confirm that the qmake.exe is the new one:

* Type:  C:\RiverWare\tools\UnxUtils\which qmake.exe
* See Result:  C:\Riverware\tools\Qt-484\bin\qmake.exe

(6) Generate the Visual Studio ".vcxproj" files (from Qwt's ".pro" files):

* Open the desired Visual Studio Command Prompt (32-bit or 64-bit; See Qt step 6, above).
* Change to directory:  C:\Riverware\tools\Qwt-522\_Qt-484\
* Type:  admin\msvc-qmake vc

Among other things, this results in the creation of file: src\qwtd.vcxproj. Note: the 'd' signifies 'debug' (rather than 'release'). But we really don't need a separate vcxproj file for 'release'. That can just be toggled within Visual Studio.

(7) In Visual Studio 2010, open the project file:

* C:\Riverware\tools\Qwt-522\_Qt-482\src\qwtd.vcxproj

(7a) Build the Release (non-debug) and Debug Libraries:

1. Build >> Clean Solution
2. Build >> Build Solution ... acknowledge the saving of the solution (.sln) file.
3. Change from Release to Debug (or Debug to Release -- *the order doesn't matter*).
4. Build >> Build Solution

(8) For the 64-bit build, insure that the two generated libraries are in, or moved to C:\RiverWare\tools\Qwt-522\_Qt-482\lib\. This is the normal destination for the 32-bit libraries, and I have also seen the 64-bit build put them there too -- but not always. That is, the 64-bit libraries may be created here:

* C:\RiverWare\tools\Qwt-522\_Qt-482\src\x64\Release\qwt.lib
* C:\RiverWare\tools\Qwt-522\_Qt-482\src\x64\Debug\qwtd.lib

If so, copy them here:

* C:\RiverWare\tools\Qwt-522\_Qt-482\lib\qwt.lib
* C:\RiverWare\tools\Qwt-522\_Qt-482\lib\qwtd.lib

## ClientServer

ClientServer is a CADSWES-created library that contains common code used in RiverWare and in the database DMI servers to enable client-server communication between RiverWare and the servers. This library used to be in a stand-alone CVS repository and was built via a Visual Studio solution and distributed to development machines in the \RiverWare\tools directory as part of the win-config.pl process. It has now been incorporated into the RiverWare git repository and appears as a Visual Studio project under the RiverWare Visual Studio solution. The ClientServer library is built as part of the RiverWare build and is no longer distributed to development machines under \RiverWare \tools.

There is a Visual Studio solution under the ClientServer directory that can be used to build a test executable for server connectivity. This solution isn’t normally built as part of RiverWare, but can be utilized for testing and diagnosing server problems.

## CPLEX

CPLEX is the licensed optimization software from IBM that is used with RiverWare optimization. Tim can download the CPLEX installation packages from the IBM website. Following are steps for getting our CPLEX files using version 12.5 as an example on 32 bit.

(1) Run the CPLEX installer (i.e. cplex\_studio125.win-x86-32.exe). This will install CPLEX Optimization Studio into C:\Program Files\IBM\ILOG\CPLEX\_Studio125.

(2) Copy the concert include directory from:

* C:\Program Files\IBM\ILOG\CPLEX\_Studio125\concert\include

to

* C:\RiverWare\tools\CPLEX-125\concert\include.

(3) Copy concert.lib from:

* C:\Program Files\IBM\ILOG\CPLEX\_Studio125\concert\lib\x86\_windows\_vs2010\stat\_mda

to

* C:\RiverWare\tools\CPLEX-125\concert\lib\release

(4) Copy concert.lib from:

* C:\Program Files\IBM\ILOG\CPLEX\_Studio125\concert\lib\x86\_windows\_vs2010\stat\_mdd

to

* C:\RiverWare\tools\CPLEX-125\concert\lib\debug

(5) Copy the cplex include directory from:

* C:\Program Files\IBM\ILOG\CPLEX\_Studio125\cplex\include

to

* C:\RiverWare\tools\CPLEX-125\cplex\include

(6) Copy all of the cplex libraries (there are 10 in version 12.5) from

* C:\Program Files\IBM\ILOG\CPLEX\_Studio125\cplex\lib\x86\_windows\_vs2010\stat\_mda

to

* C:\RiverWare\tools\CPLEX-125\cplex\lib\release

(7) Copy all of the cplex libraries (there are 10 in version 12.5) from

* C:\Program Files\IBM\ILOG\CPLEX\_Studio125\cplex\lib\x86\_windows\_vs2010\stat\_mdd

to

* C:\RiverWare\tools\CPLEX-125\cplex\lib\debug

(8) Copy the cplex125.dll from

* C:\Program Files\IBM\ILOG\CPLEX\_Studio125\cplex\bin\x86\_win32

to

* C:\RiverWare\tools\bin

## GDAL

GDAL is the geospatial library that is used in implementing map backgrounds in RiverWare’s geospatial view of the workspace. GDAL is built with the MRSID SDK, which provides support for the proprietary MRSID and the JPEG2000 image formats for RiverWare. Instruction below give steps for installing MRSID and then installing and building GDAL.

### MRSID Software Development Kit

(1) Register with the LizardTech website: [www.lizardtech.com](http://www.lizardtech.com) . They will send you a password via email.

(2) Log in to the website and go to the Developer home page by following the menu links ABOUT\_US->DEVELOPERS->DEVELOPER HOME.

(3) Go to the Download SDKs tab and download the zip file for the appropriate SDK (i.e. MrSID\_DSDK-8.5.0.3422-win32-vs10.zip).

(4) Unzip the SDK into a local directory. This location will be referenced when GDAL is built.

### Building GDAL

Primary instruction for building GDAL can be found at:

* <http://trac.osgeo.org/gdal/wiki/BuildingOnWindows>

Supplementary instructions for installing GDAL/OGR as a library to be linked in (dynamically) with RiverWare follow using gdal-1.9.0 as an example:

(1) Get the GDAL source code and put it in an appropriate location. For example, download the source code in zip file form from:

* http://trac.osgeo.org/gdal/wiki/DownloadSource

Then extract it to somewhere, not the tools directory, e.g.:

* C:\RiverWare\staff\lynn

The resulting directory in this case is gdal-1.9.0, which I renamed to "gdal-1.9.0 (full)" to avoid confusion with the tools directory.

(2) Make a copy of nmake.opt in the source directory and edit it to set the appropriate compilation options.

Set the home directory:

* GDAL\_HOME: "C:\RiverWare\tools\gdal-1.9.0"

For a 64-bit build, uncomment the following line:

* WIN64=YES

To include MRSID, uncomment the following lines and set the MRSID\_DIR path to your local directory where the MRSID SDK is located (see MRSID Software Development Kit step 4 above).

* MRSID\_DIR = C:\RiverWare\staff\lynn\MrSID\_DSDK-8.5.0.3422-win64-vs10
* MRSID\_JP2 = YES

By default a release version is built; for a debug version I think one would need to define DEBUG:

(3) Create the appropriate tools directory:

* C:\RiverWare\tools\gdal-1.9.0

(4) Compile GDAL

Open one of the two Visual Studio 2010 command prompts, depending on whether a 32-bit or 64-bit build of Qt is desired. From the Windows Start Menu: All Programs >> Microsoft Visual Studio 2010 >> Visual Studio Tools >> ...

* *32-bit Build:*  Visual Studio Command Prompt (2010)
* *64-bit Build:*  Visual Studio x64 Win64 Command Prompt (2010)

Change to the GDAL directory and then compile (this takes about 5 minutes on my machine):

* nmake -f makefile.vc > build.out

(5) In the same shell, install the library:

* nmake -f makefile.vc install > install.out

This builds GDAL utilities (standalone executables), puts them in the bin directory, and puts data in the data directory. In particular, this creates the following directories:

* C:\RiverWare\tools\gdal-1.9.0\bin
* C:\RiverWare\tools\gdal-1.9.0\data
* C:\RiverWare\tools\gdal-1.9.0\html

The system issues many messages indicating that a file cannot be found; this does not appear to be a problem.

(6) For development, we need to install the include and lib directories:

* nmake -f makefile.vc devinstall > devinstall.out

This creates the following directories:

* C:\RiverWare\tools\gdal-1.9.0\lib
* C:\RiverWare\tools\gdal-1.9.0\include

The necessary include files and gdal\_i.lib are placed there.

The system issues many messages indicating that a directory already exists or a file cannot be found; this does not appear to be a problem.

(7) Copy the GDAL dll (gdal19.dll) from the gdal-1.9.0(full) directory to C:\RiverWare\tools\bin

(8) Copy the lti\_dsdk.dll from the MrSID SDK directory (C:\RiverWare\staff\lynn\MrSID\_DSDK-8.5.0.3422-win64-vs10\Raster\_DSDK\lib) to C:\RiverWare\tools\bin.

(9) Copy the lti\_lidar\_dsdk.dll from the MrSID SDK directory (C:\RiverWare\staff\lynn\MrSID\_DSDK-8.5.0.3422-win64-vs10\Lidar\_DSDK\lib) to C:\RiverWare\tools\bin.

(10) PROJ.4 is a cartographic projections library whose dll is used in RiverWare as part of the GDAL functionality. Find an appropriate copy of its proj.dll and place it in C:\RiverWare\tools\bin.

In one case, I got this dll by downloading a GDAL installer and installing GDAL in:

* C:\Program Files\GDAL

In addition to installing GDAL, this installation places proj.dll in that directory.

In another case, I downloaded the proj4 binaries direcly from:

* http://trac.osgeo.org/proj/

## Reprise

Reprise is the license management software used for RiverWare. Jessica downloads new versions and builds them with Visual Studio. The pieces we need are listed as followsusing Reprise 10.0 as an example:

(1) Copy the following files from the Reprise installation to C:\RiverWare\tools\RepriseLM-10.0:

* license.h
* rlm1002.dll
* rlm1002.lib
* rlmid1.lib

(2) Copy the rlm1002.dll from the Reprise installation to C:\RiverWare\tools\bin.

## RdfToExcelLib

The RdfToExcelLib is a CADSWES-created library that is statically linked to RiverWare to provide the capability to create and write Excel files from RDF files. All of the Excel-writing code is contained in the Study Manager GIT repository. This code is used for both the Study Manager and for RiverWare, but is kept in a single place for ease of maintenance. Steps for building RdfToExcelLib follow:

(1) Use Git Extensions to clone a local copy of the Study Manager repository located at [\\alamosa\projects\StudyMgr\StudyMgr.git](file:///\\alamosa\projects\StudyMgr\StudyMgr.git). Let’s assume the clone was put into C:\Riverware\staff\nwilson\StudyMgr.

(2) Run qmake on RdfToExcelLib.pro in C:\Riverware\staff\nwilson\StudyMgr\RdfToExcelLib to generate the Visual Studio project file.

(3) Open RdfToExcelLib.sln located in this directory with Visual Studio and build the debug and release libraries.

(4) Copy header files to the appropriate RiverWare tools directory (i.e. C:\Riverware\tools\RdfToExcel\_Qt-484\include).

* From C:\Riverware\staff\nwilson\StudyMgr\ExcelWrite\ copy the following:
  + ExcelWrite.hpp
  + ExcelAddIn.hpp
* From C:\Riverware\staff\nwilson\StudyMgr\Utils\ copy the following:
  + Okstat.hpp
* From C:\Riverware\staff\nwilson\StudyMgr\RDF\ copy the following:
  + RdfFile.hpp
  + RdfUtils.hpp

(5) Copy library files to the appropriate RiverWare tools directory (i.e. C:\Riverware\tools\RdfToExcel\_Qt-484\lib).

* Copy:
  + C:\RiverWare\staff\nwilson\StudyMgr\RdfToExcelLib\release\RdfToExcelLib.lib

and paste with a new name as:

* + C:\RiverWare\tools\RdfToExcel\_Qt-484\lib\RdfToExcelLib-md.lib
* Copy:
  + C:\RiverWare\staff\nwilson\StudyMgr\RdfToExcelLib\debug\RdfToExcelLib.lib

and paste with a new name as:

* + C:\RiverWare\tools\RdfToExcel\_Qt-484\lib\RdfToExcelLib-mdd.lib
* Copy:
  + C:\RiverWare\staff\nwilson\StudyMgr\RdfToExcelLib\debug\rdftoexcellib.pdb

to:

* C:\RiverWare\tools\RdfToExcel\_Qt-484\lib\rdftoexcellib.pdb

## Tcl

Tcl-851 is currently required for linking into RiverWare. This library did not have to be rebuilt for the Visual Studio 2010 port. When it does need to be rebuilt or upgraded, the procedure should be documented here.

## NetCDF

NetCDF libraries are used for writing netCDF format files from multiple runs. Pre-built netCDF libraries for Visual Studio (compatible with VS 2010) are available at:

<http://www.unidata.ucar.edu/software/netcdf/docs/winbin.html>

Packages for netCDF4 were downloaded and used for RiverWare (netCDF4.3.2-NC4-32.exe and netCDF4.3.2-NC4-64.exe). These executables are install packages that will install the netCDF libraries (install into C:\RiverWare\tools\netCDF-4.3.2, for example). The bin, include and lib directories under the installation contain the netCDF files necessary for RiverWare. Also under the deps subdirectory, there are additional bin, include, and lib directories that contain files needed by netCDF for zlib, hdf5, and hdf\_hl. The dlls for these and netCDF must be copied to C:\Riverware\tools\bin and also be included in the RiverWare release package (netcdf.dll, hdf5.dll, hdf5\_hl.dll, and zlib.dll).

Note that quaZIP is also referencing the zlib files that are in the netCDF directory structure in RiverWare\tools. If the version of netCDF is updated and its RiverWare\tools directory is renamed, the inclusion in the Quazip.pro file will need to be updated and quaZip rebuilt.

## QuaZIP

QuaZIP is used for zipping an unzipping RiverWare package files. The program is described at <http://quazip.sourceforge.net/> and can be downloaded at <http://sourceforge.net/projects/quazip/>.

As of this writing, version 0.7 of QuaZIP was built as a dynamically linked library for use with Riverware. QuaZIP uses Qt, so the library will need to be rebuilt when Qt versions are changed. The .pro file that came with the QuaZIP download was not used, but a more straightforward one was created (Quazip.pro) that will create release and debug libraries and dlls with different names. Steps for building the libraries (using version 0.7 as an example) are described below:

(1) Copy the source files from the quazip-0.7\quazip subdirectory to the directory where you want to do the new build.

(2) Copy the Quazip.pro file that is CADSWES-created from the C:\Riverware\tools\Quazip-0.7\ directory. Also copy the quazipd.sln file.

(3) Run qmake from the current version of Qt on Quazip.pro to create an updated Visual Studio project file (quazipd.vcxproj).

(4) Open the quazipd.sln file in Visual Studio and build the debug and release configurations. This will create quazipd.lib and quazipd.dll, and quazip.lib and quazip.dll in the lib subdirectory.

Note that QuaZIP needs zlib.lib. We use the zlib version that is part of our netCDF library directory by referring to it in the quazip.pro file, and the dll for this version of zlib is included with RiverWare. If you are running RiverWare in a development environment, it can cause a RiverWare crash if a different version of zlib.dll is being picked up in your path before the one in C:\Riverware\tools\bin.

## QsLog

QsLog is used for messaging across threads and is employed in RiverSMART and in the distributed MRM controller.

As of this writing, version 2.0b3 of QsLog was built as a dynamically linked library for use with RiverSMART and RiverWare. QsLog uses Qt, so the library will need to be rebuilt when Qt versions are changed. QsLog code is contained in the C:\Riverware\tools\QsLog directory and is part of the win-config.pl process. The QsLog.pro file was created, which will result in release and debug libraries and dlls with different names. Steps for building the libraries are described below:

(1) Run qmake from the current version of Qt on QsLog.pro to create an updated Visual Studio project file (QsLogd.vcxproj).

(4) Open the QsLog.sln file in Visual Studio and build the debug and release configurations. This will create QsLogd.lib and QsLogd.dll, and QsLog.lib and QsLog.dll in the debug and release subdirectories, respectively.

(5) Copy the two dlls into C:\Riverware\tools\bin

# Associated Executables

This section covers other executables that we ship with RiverWare.

## Remote

The Remote solution is located in the RiverWare repository under Remote\Remote. It generates one executable that is used in the distributed MRM functionality: RwRemoteMgr.exe. To build under Visual Studio, run qmake on RwRemoteMgr.pro, open the Remote.sln with Visual Studio, and rebuild.

Note that there is also a CombineRdf.pl file in the Remote\Remote directory that is a Perl script executable used to combine RDF files from distributed multiple runs. This must be included in the RiverWare release package.

## Servers

The Server executables were used in the ClientServer paradigm to connect RiverWare to other software or data repositories (HDB, DSS, Modflow). Code for the servers was located in the RiverWare repository under Servers. As of this writing, we are currently not building any of these for RiverWare releases. The Modflow connection options in RiverWare have been removed or made unavailable so the Modflow server is obsolete. Due to its Fortran library’s incompatibility with the Microsoft C++ libraries used in our build of Qt4, the DSS server is being replaced with a java-based server application. The HDB server has been eliminated and the code for connecting and communicating with Oracle and HDB has been moved into RiverWare.

# Other Executables

This section covers other CADSWES executables that we do not ship with RiverWare.

## RdfToExcelExecutable

The RdfToExcelExecutable is a stand-alone executable version of the Rdf to Excel code (this replaces the old ExcelWriter executable). The RDF to Excel conversion code is maintained in the Study Manager GIT repository and is used for both the Study Manager and for RiverWare in addition to this stand-alone executable. It is kept in a single place for ease of maintenance. Steps for building RdfToExcelLibExecutable follow:

(1) Use Git Extensions to clone a local copy of the Study Manager repository located at [\\alamosa\projects\StudyMgr\StudyMgr.git](file:///\\alamosa\projects\StudyMgr\StudyMgr.git). Let’s assume the clone was put into C:\Riverware\staff\nwilson\StudyMgr.

(2) Run qmake on RdfToExcelExecutable.pro in C:\Riverware\staff\nwilson\StudyMgr\RdfToExcelExec to generate the Visual Studio project file.

(3) Open RdfToExcelExecutable.sln located in this directory with Visual Studio and build the debug and release executables.

(4) The release executable is posted to the RiverWare download page on the riverware.org website

## RdfAnnualizerExecutable

The RdfAnnualizerExecutable is a stand-alone executable version of the code to aggregate an RDF file of shorter timestep to an annual RDF file (this replaces the old Yearly Aggregation Post Processor). The RDF annualizer code is maintained in the Study Manager GIT repository and is used for a Study Manager plugin in addition to the stand-alone executable. It is kept in a single place for ease of maintenance. Steps for building RdfAnnualizerExecutable follow:

(1) Use Git Extensions to clone a local copy of the Study Manager repository located at [\\alamosa\projects\StudyMgr\StudyMgr.git](file:///\\alamosa\projects\StudyMgr\StudyMgr.git). Let’s assume the clone was put into C:\Riverware\staff\nwilson\StudyMgr.

(2) Run qmake on RdfAnnualizerExecutable.pro in C:\Riverware\staff\nwilson\StudyMgr\ RdfAnnualizerExec to generate the Visual Studio project file.

(3) Open RdfAnnualizerExecutable.sln located in this directory with Visual Studio and build the debug and release executables.

(4) The release executable is posted to the RiverWare download page on the riverware.org website

## Graphical Policy Analysis Tool

The Graphical Policy Analysis Tool (GPAT) is an add-in to Excel written in Visual Basic for Applications (VBA). As such, its code resides in the Excel add-in file (i.e. Graphical Policy Analysis Tool 3.xla). In order to access the code, the .xla file must be loaded as an add-in to Excel under the File/Options/Add-Ins menu. Code can then be found in the Visual Basic editor under the Developer menu. The code is password protected, so the password must be entered when the code is expanded in the Visual Basic Editor window. The password is created in the properties dialog (right click on the project) and is typically changed with each major release version. A file called GpatPasswords.txt is maintained to contain the passwords for all the versions. A document called GpatProgrammerDoc.doc describes the general structure of the GPAT code base. Once code is changed in the Visual Basic editor, it is compiled using the Debug/Compile VBA Project menu item, which saves changes back into the loaded .xla file. GPAT is posted for users to download on the RiverWare download page on the riverware.org website.

GPAT had been a project under the RiverWare CVS repository, but was not moved to the RiverWare git repository. It was decided at that time to put GPAT into the Study Manager git repository, but this has not happened yet. In the meantime, the current versions of the GPAT files are saved under projects/riverware/doc/GPAT.

## Demand Input Tool

The Demand Input Tool (DIT) is an add-in to Excel written in Visual Basic for Applications (VBA). As such, its code resides in the Excel add-in file (i.e. DemandInputTool.xlam). In order to access the code, the .xlam file must be loaded as an add-in to Excel under the File/Options/Add-Ins menu. Code can then be found in the Visual Basic editor under the Developer menu. The code is password protected, so the password must be entered when the code is expanded in the Visual Basic Editor window. The password is created in the properties dialog (right click on the project). A file called DemandInputToolPasswords.txt is maintained to contain the passwords the versions. Once code is changed in the Visual Basic editor, it is compiled using the Debug/Compile VBA Project menu item, which saves changes back into the loaded .xlam file. DIT is posted for users to download on the waterSMART web page.

The DIT had been a project under the RiverWare CVS repository, but was not moved to the RiverWare git repository. It was decided at that time to put DIT into the Study Manager git repository, but this has not happened yet. In the meantime, the current versions of the DIT files are saved under projects/riverware/doc/DemandInputTool.