Building Qt 5.4.2 and ICU / CADSWES / June 2015

Phil Weinstein, Original Draft: 6-9-2015. Edit: 6-10-2015 (a).
Document home: R:\doc\Qt\2015\BuildingQt5-June2015.html (and .pdf).

Overview

This document describes the process of acquiring and building Qt 5.4.2 (MSVC 2010, 64-bit, debug and release) and supporting tools for the purpose of porting RiverWare and other CADSWES Qt applications from Qt 4.8.5 to Qt 5.

This includes:

- Building **ICU** from source code.
- Building **Qt 5.4.2** from source code.
- Inclusion of **Qt WebKit** in Qt5.

This lacks:

- Inclusion of **Qt WebEngine** in Qt5. This isn't quite ready for use. We will probably want to wait for Qt 5.6 -- i.e. *two* minor versions in the future -- before attempting to build and use Qt WebEngine. Also, Microsoft Visual Studio builds of Qt WebEngine will *officially* require Visual Studio 2013. (We're currently using Visual Studio 2010).
- Inclusion of special graphics driver modules (needed for Qt Quick 2): **DirectX, OpenGL, ANGLE,** etc. These don't seem to be necessary for our Qt4 to Qt5 porting effort. (We'll know for sure when we can link and run RiverWare). The way these are integrated into the Qt build is changing in the upcoming Qt 5.5.0 release (planned for late June 2015).

Contents:

Document Status / Revision History:

- 6-10-2015: Added *Possible Improvements;* other revisions.
- 6-09-2015: Initial Writing.

Related CADSWES Documents:

Once the ICU and Qt 5 builds become part of our development tools, the following document should be revised to reflect information in this document:

- **Building Libraries and Executables Supporting RiverWare**
  R:\windows\config\BuildingRiverWareRelatedSoftware.docx
  Most recent revision: 10-24-2014.

The following Qt reference pages are useful:

- Qt for Windows - Requirements
  http://doc.qt.io/qt-5/windows-requirements.html
- Qt for Windows - Building from Source
  http://doc.qt.io/qt-5/windows-building.html
- Qt Configure Options
  http://doc.qt.io/qt-5/configure-options.html
- Build Instructions for the QtWebKit build on Windows
  http://trac.webkit.org/wiki/BuildingQtOnWindows
  ... note that we are using x64 command line Visual Studio tools, not MinGW, nor Cygwin.

Also, I have put aside these two README and HELP files, here:

- HOW TO BUILD QT5 (from Qt 5.4.2 source distribution)
  http://cadswes2.colorado.edu/~philw/2015/Qt5/notes/README.txt
- Qt build "configure" script HELP output
  http://cadswes2.colorado.edu/~philw/2015/Qt5/notes/configure_help.txt
- International Components for Unicode / ICU 55.1 ReadMe
  http://cadswes2.colorado.edu/~philw/2015/Qt5/notes/ics551-readme.html

Possible Improvements to our Qt5 Build Implementation

1. The invocation of the Qt build "configure" script -- with the important list of switches (command line arguments) should be encapsulated in a saved configuration. See the `-saveconfig` and `-loadconfig` parameters to "configure."
2. Build Environment Variables: Instead of adding the necessary additional values to the actual *path, include,* and *lib* environment variables, for some of these, it would be better to put that into a Windows command script -- to set those definitions for only the *build session.* SEE "Step 3" in the *"Building from Source"* webpage cited above. (It may still be desirable to add the ICU bin64 folder to the actual system path environment variable -- RiverWare will need those DLLs too).
3. It would be useful to look into "jom" -- a Qt multi-process equivalent to "nmake." Faster. Jom *is* officially recommended (as is nmake) in the Qt5 build instructions. It's possible that "jom" can be built from this Qt5 distrubution. In any case, it's available. See Introduction to jom webpage [https://wiki.qt.io/Jom].
4. The "How to Build Qt5" README file recommends checking out these utilities:
   - The submodule repository `qtrepotools` contains useful scripts for developers and release engineers. Consider adding `qtrepotools/bin` to your PATH environment variable to access them.
   - The `qt5_tool` in `qtrepotools` has some more features which may be of interest. Try `'qt5_tool --help'`.

(I) General Libraries -- just download and install

1. ActivePerl (5.12 or later) (I used 5.20.2) (See **Perl Conflict** note).
2. Python 3.4 (I used 3.4.0)
3. Ruby version 1.9.3

I won't here provide comprehensive instructions for these installations. We don't need to build these from source code -- just the normal binary installations for Windows are used.

Those three components are installed at these locations:

Installation Directory	Added to Path:	Where*
C:\Perl64\	C:\Perl64\site\bin;C:\Perl64\bin;	beginning
C:\Python34\	C:\Python34\;C:\Python34\Scripts;	beginning
C:\Ruby193\	C:\Ruby193\bin;	end

*This is just where I had these on my path for my successful Qt 5.4.2 build.

Perl Conflict: Unfortunately, Qt requires a more advanced version of **Perl,** version 5.12 or later, than for which Perl/Tk is available. We need to stick with Perl 5.6.1 to support our internal tools using Perl/Tk, e.g. the **RiverWare regression tests.** (Newer Perl versions lack Perl/Tk because a maintainer was no longer available). I installed a 64-bit version of Perl 5.20.2 -- just to build Qt.

- Note: After completing my Qt5 builds, I was unable to UN-install Perl 5.20.2 using the administrator account which I have access to on my machine (.\Gomer Piles). So, for now, I can't run RiverWare regression tests on this machine.

Python: The Qt5 build technically requires only Python 2.7 or later. Even though Python 2 and Python 3 are slightly different languages, Python 3 (version 3.4.0) apparently does work for the Qt5 build. We use Python 3.4 for processing **RPL Predefined Function help content** for RiverWare. (And I believe all of our Windows development machines currently have this). See the following document:

- **Processing RPL Predefined Function Help for use in the RiverWare Build**
  R:\doc\RPL\EmbedRplDoc\2015\RplFuncHelpInRiverWare-March2015.docx
  R:\doc\RPL\EmbedRplDoc\2015\RplFuncHelpInRiverWare-March2015.pdf
  http://cadswes2.colorado.edu/~philw/2015/RplDoc/Process/GenRplFuncContent.html

Notably, this Qt 5.4.2 build does NOT make use of the following 3rd party graphics driver libraries, required for Qt Quick 2 / QML:

1. OpenGL
2. ANGLE
3. DirectX 11 or DirectX 9

(II) ICU (International Components for Unicode) -- version 55.1 or later

> "ICU (International Components for Unicode) is a mature, widely used set of C/C++ and Java libraries providing Unicode and Globalization support for software applications. ICU is widely portable and gives applications the same results on all platforms and between C/C++ and Java software. ICU is released under a nonrestrictive open source license that is suitable for use with both commercial software and with other open source or free software." [http://site.icu-project.org/].

Having ICU installed, and distributing ICU run time components, is a requirement for Qt WebKit in Qt5. It is also used by other Qt components, *when present,* and is expected to become a broader requirement (i.e. not just for Qt WebKit) in subsequent Qt5 versions.

ICU will be added as a standard component of our RiverWare build system. The following are instructions for obtaining and building ICU from source.

(II.A) Download ICU 55.1 (or later) source tree

1. Go to http://site.icu-project.org/download
2. Click the latest version if ICU4C (i.e. *for C/C++,* not Java) ... e.g. ICU4C / 55.1
3. Scroll down to **ICU4C Source Code Download** -- i.e. *don't use the Binary download.*
4. Click on: `icu4c-55_1-src.zip` ... **ZIP file for Windows platforms** ... (25.8 MB) ... Save File ... to an arbitrary good place on your machine.

(II.B) Extract content from of icu4c-55_1-src.zip

Any ZIP file extraction tool should do. I have 7-Zip installed on my Windows 7 machine.

- Move aside (rename) any existing `C:\Riverware\tools\icu-55.1` folder.

- *[assuming 7-Zip is being used]* ...
  a. Right-click on the downloaded `icu4c-55_1-src.zip` file ...
  b. 7-Zip >> Extract files ...
  c. Extract to: `C:\Riverware\tools\`
  d. Rename the resulting new folder in `C:\Riverware\tools\` from `icu` to `icu-55.1`.
     - **Note:** This path must not contain any spaces or Windows specific file system characters.

The resulting folder, `C:\Riverware\tools\icu-55.1`, has the full ICU source tree. We will be building that in place. At this point (*before the build*) this folder has the following statistics. (Right click on the folder name, and select Properties ...):

- Size: 107 MB (112,663,468 bytes)
- Size on Disk: 118 MB (124,649,472 bytes)
- Contains: 4,505 Files, 125 Folders

Browse the C:/Riverware/tools/icu-55.1/readme.html webpage.

Even though the **bin64** directory under that ICU folder does not yet exist, the instructions call for adding that bin directory to your system PATH before building ICU. Note that that readme.html file refers to this as just the "bin" directory. But for 64-bit, it is bin64. *You might as well also make similar additions to the include and lib environment variables at this point.*

So, in your Control Panel >> System >> Advanced system settings (requires an administrator account, e.g. `.\Gomer Piles`, and password) ... Environment Variables ... System variables:

System Environment Variable	Value
ADD to Path:	C:\Riverware\tools\icu-55.1\bin64
ADD to (or create) include:	C:\Riverware\tools\icu-55.1\include
ADD to (or create) lib:	C:\Riverware\tools\icu-55.1\lib64

The remaining instructions in the C:/Riverware/tools/icu-55.1/readme.html webpage -- *How To Build and Install on Windows* **-- are pretty clear.**

There is a Microsoft Visual Studio "all in one" solution (SLN) file which works nicely in Visual Studio 2010: `C:\Riverware\tools\icu-55.1\source\allinone\allinone.sln`.

- It's apparently fine for the first project, "cal", to be selected as the StartUp project, (i.e. bolded).
- Build both 64-bit Debug and Release builds:
  - ... Select "x64"
  - ... Debug (... Rebuild Solution)
  - ... Release (... Build Solution).
- There are compilation warnings. Apparently OK.

Since the environment variables are all set up (done above), you should be able to run ICU's self tests:

- `C:\Riverware\tools\icu-55.1\source\allinone\icucheck.bat x64 Debug`
- `C:\Riverware\tools\icu-55.1\source\allinone\icucheck.bat x64 Release`

At this point, the `C:\Riverware\tools\icu-55.1` folder has the following statistics. (Right click on the folder name, and select Properties ...):

- Size: 767 MB (805,262,055 bytes)
- Size on Disk: 797 MB (836,218,880 bytes)
- Contains: 12,052 Files, 310 Folders

(III) Qt 5.4.2

(III.A) Obtain and Install your Qt Commercial License File (four of our developers have these).

1. Log on to https://login.qt.io/login (my login e-mail address is Philip.Weinstein@colorado.edu).
2. Navigate to Licenses and Subscriptions ... click Download Qt License ... Save File:
   - original file name is `qt-license.txt`
3. Copy that file to your home directory and rename (*using command prompt*):
   - Your home directory is typically the value of the USERPROFILE environment variable.
   - ... type "set" in a command line window and look for USERPROFILE.
   - ... for me that's C:\Users\philw
   - Rename (*using command prompt!*) to "`.qt-license`" (notice the leading period).

   NOTE: An alternative to having the license file in the proper place is to type in the following information from that license file when prompted by the "configure" script (later on in the build process, see below). The following two lines from the license file are required: (a) **Licensee** (for me that's "Weinstein Phil", without the quotes), and (b) **Qt5LicenseKey** (that looks like: XXXX-XXX-XXX-XXX-XXXXX-XXXXX-XXXX).

(III.B) Download the Qt5 Source zip file

1. Log on to https://login.qt.io/login (my login e-mail address is Philip.Weinstein@colorado.edu), as above.
2. Navigate to Downloads
3. Select Product: Qt
4. Select Version. I picked the latest available: 5.4.2.
5. Of the 13 "Results" choices, pick this one:
   - **Qt Source Package, Full Framework with Windows style line endings 5.4.2**
   - Filename: `qt-everywhere-enterprise-src-5.4.2.zip`

6. Click the "Download" button ... Save File ... to an arbitrary good place on your machine.

(III.C) Extract content from qt-everywhere-enterprise-src-5.4.2.zip

Any ZIP file extraction tool should do. I have 7-Zip installed on my Windows 7 machine.

1. Move aside (rename) any existing `C:\Riverware\tools\Qt-542` folder

2. *[assuming 7-Zip is being used]* ...
   a. Right-click on the downloaded `qt-everywhere-enterprise-src-5.4.2.zip` file ...
   b. 7-Zip >> Extract files ...
   c. Extract to: `C:\Riverware\tools\`
   d. Rename the resulting new folder in `C:\Riverware\tools\` from `qt-everywhere-enterprise-src-5.4.2` to `Qt-542`.
      - **Note:** This path must not contain any spaces or Windows specific file system characters.

The resulting folder, `C:\Riverware\tools\Qt-542`, has the full Qt source tree. We will be building that in place. At this point (*before the build*) this folder has the following statistics. (Right click on the folder name, and select Properties ...):

- Size: 1.48 GB (1,599,616,018 bytes)
- Size on Disk: 1.79 GB (1,926,406,144 bytes)
- Contains: 132,681 Files, 13,546 Folders.

(III.D) Complete Qt Build Environment Setup

The Qt5 build (in particular for Qt WebKit) also requires **Bison, GPerf,** and **Flex.** Versions of these GNU utilities are part of the Qt5 source code distribution, in a single directory. All that we need to do for these is to add `C:\Riverware\tools\Qt-542\gnuwin32\bin` to the end of the path.

As mentioned in prior sections, the **path, include,** and **lib** system environment variables are accessible from: Control Panel >> System >> Advanced system settings (requires an administrator account, e.g. `.\Gomer Piles`, and password) ... Environment Variables ... System variables. The following table reflects the content of these variables specifically used by the Qt5 build:

System Environment Variable	Value

Path (towards beginning)	`C:\Perl64\site\bin;`
	`C:\Perl64\bin;`
	`C:\Python34\;`
	`C:\Python34\Scripts;`

Path (towards the end)	`C:\Ruby193\bin;`
	`C:\Riverware\tools\icu-55.1\bin64;`
	`C:\Riverware\tools\Qt-542\gnuwin32\bin;`

include	`C:\Riverware\tools\icu-55.1\include`
lib	`C:\Riverware\tools\icu-55.1\lib64`

ALSO: It is (or, at least, may be) important to **REMOVE** our Unix Utilities from the path during the Qt5 Build Configuration and Build (nmake) process:

System Environment Variable	REMOVE Value
Path	`C:\RiverWare\tools\UnxUtils;`

(III.E) Configure the Qt5 Build

This is done by executing the Qt5 **"configure"** script within a **Visual Studio x64 Win64 Command Prompt (2010).** On a machine with Visual Studio 2010 installed, this command window is available from the Start Menu >> All Programs >> Microsoft Visual Studio 2010 >> Visual Studio Tools >> Visual Studio x64 Win64 Command Prompt.

Open one of these up, and type:

- `C:`
- `cd \Riverware\tools\Qt-542`
- `path` ... *to confirm the path, if you'd like.*

Very big deal: Carefully type (copy and paste) this -- *all one line:*

- **`configure -prefix %CD%\qtbase -commercial -nomake tests -debug-and-release -commercial -platform win32-msvc2010 -shared -qt-zlib -qt-libpng -qt-libjpeg -no-opengl -no-angle -icu`**

NOTE: The following comment generated from "configure" -- *"To reconfigure, run nmake confclean and configure"* -- turns out to be incorrect; this is not actually supported in Qt5. (It's apparently left over from Qt4). To redo the configuration, I ended up unpacking the original source into a fresh `C:\Riverware\tools\Qt-542` build directory.

NOTE: The configure command for our prior Qt build (for Qt 4.8.5) also had the following switches which are no longer used in Qt5. When we subsequently do a production build of Qt5 -- i.e. not just for the purpose of Qt4-to-Qt5 porting work -- we will want to review the full set of configure switches used -- and not used. I'm particularly concerned about the removal of the `-no-exceptions` switch.

- NOT USED FOR QT5: `-no-exceptions -qt-libmng -no-libtiff -webkit`

NOTE: The "`-no-opengl -no-angle`" switches were added to resolve the following error message, generated from "configure" -- see following. (I believe we don't currently need OpenGL support).

- The DirectX SDK could not be detected: There is no Direct X SDK installed or the environment variable "DXSDK_DIR" is not set. Disabling the ANGLE backend. WARNING: Using OpenGL ES 2.0 without ANGLE. Specify -opengl desktop to use Open GL. *The build will most likely fail.*

If your Qt Commercial license isn't installed -- *see step III-A* -- the configure script will prompt for the following information (my responses are underlined):

- Licensee: `Weinstein Phil`
- Qt5 License Key: `xxxx-xxx-xxx-xxx-xxxxx-xxxxx-xxxx` (available from https://account.qt.io/)

Running configure (*see above*) took about only one minute. The following is the last part of the output to the console:

```
... ... ...
Microsoft (R) Incremental Linker Version 10.00.40219.01
```

```
Copyright (C) Microsoft Corporation.  All rights reserved.

     copy qmake.exe C:\Riverware\tools\Qt-542\qtbase\bin\qmake.exe
     1 file(s) copied.

Running configuration tests...
Environment:

   INCLUDE=
     C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\INCLUDE
     C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\ATLMFC\INCLUDE
     C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\include
     C:\Riverware\tools\icu-55.1\include

   LIB=
     C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\LIB\amd64
     C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\ATLMFC\LIB\amd64
     C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\lib\x64
     C:\Riverware\tools\icu-55.1\lib64

   PATH=
     C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\BIN\amd64
     C:\Windows\Microsoft.NET\Framework64\v4.0.30319
     C:\Windows\Microsoft.NET\Framework64\v3.5
     C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\VCPackages
     C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE
     C:\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\Tools
     C:\Program Files (x86)\HTML Help Workshop
     C:\Program Files (x86)\Microsoft Visual Studio 10.0\Team Tools\Performance Tools\x64
     C:\Program Files (x86)\Microsoft Visual Studio 10.0\Team Tools\Performance Tools
     C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\bin\NETFX 4.0 Tools\x64
     C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\bin\x64
     C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\bin
     C:\Perl64\site\bin
     C:\Perl64\bin
     C:\ProgramData\Oracle\Java\javapath
     C:\Python34\
     C:\Python34\Scripts
     C:\Tcl\bin
     C:\Windows\system32
     C:\Windows
     C:\Windows\System32\Wbem
     C:\Windows\System32\WindowsPowerShell\v1.0\
     C:\Program Files\Diskeeper Corporation\Diskeeper
     C:\RiverWare\bin
     C:\RiverWare\bin.reorg
     C:\RiverWare\tools\bin
     C:\RiverWare\tools\tkcvs\bin
     C:\Windows\idmu\common
     C:\Program Files (x86)\IBM\RationalPurifyPlus\Common
     C:\Program Files (x86)\Rational\common
     C:\Program Files\R\R-2.14.2\bin\x64
     C:\PROGRA~1\DISKEE~1\DISKEE~1\
     C:\Program Files (x86)\Common Files\Acronis\SnapAPI\
     C:\Program Files (x86)\Acronis\CommandLineTool\
     C:\Ruby193\bin
     C:\Riverware\tools\icu-55.1\bin64
     C:\Riverware\tools\Qt-542\gnuwin32\bin
     C:\Program Files (x86)\Common Files\Acronis\TibMounter64
```

```
Licensee....................Weinstein Phil
License ID..................
Product license.............Enterprise Edition
Expiry Date.................

Configuration:
     pcre
     debug
     compile_examples

Qt Configuration:
     minimal-config
     small-config
     medium-config
     large-config
     full-config
     debug_and_release build_all release
     debug
     shared
     zlib
     icu
     png
     freetype
     harfbuzz
     build_all
     accessibility
     audio-backend
     wmf-backend
     native-gestures
     qpa
     concurrent

     QMAKESPEC...................win32-msvc2010 (commandline)
     Architecture................x86_64, features: sse sse2
     Host Architecture...........x86_64, features: sse sse2
     Maketool....................nmake
     Debug build.................yes (combined)
     Default build...............debug
     Force debug info............no
     C++11 support...............auto
     Link Time Code Generation...no
     Accessibility support.......yes
     RTTI support................yes
     SSE2 support................yes
     SSE3 support................yes
     SSSE3 support...............yes
     SSE4.1 support..............yes
     SSE4.2 support..............yes
     AVX support.................yes
     AVX2 support................no
     NEON support................no
     OpenGL support..............no
     Large File support..........yes
     NIS support.................no
     Iconv support...............no
     Evdev support...............no
     Mtdev support...............no
     Inotify support.............no
     eventfd(7) support..........no
     Glib support................no
     CUPS support................no
     OpenVG support..............no
     SSL support.................no
     OpenSSL support.............no
     Qt D-Bus support............no
     Qt Widgets module support...yes
     Qt GUI module support.......yes
     QML debugging...............yes
     DirectWrite support.........no
     Use system proxies..........no

QPA Backends:
     GDI.....................yes
     Direct2D................no

Third Party Libraries:
     ZLIB support............qt
     GIF support.............plugin
     JPEG support............plugin
     PNG support.............yes
     FreeType support........yes
     Fontconfig support......no
     HarfBuzz support........qt
     PCRE support............qt
     ICU support.............yes
     ANGLE...................no
     Dynamic OpenGL..........no

Styles:
     Windows.................yes
     Windows XP..............yes
     Windows Vista...........yes
     Fusion..................yes
     Windows CE..............no
     Windows Mobile..........no

Sql Drivers:
     ODBC....................no
     MySQL...................no
     OCI.....................no
     PostgreSQL..............no
     TDS.....................no
     DB2.....................no
     SQLite..................plugin (qt)
     SQLite2.................no
     InterBase...............no

Sources are in..............C:\Riverware\tools\Qt-542\qtbase
Build is done in............C:\Riverware\tools\Qt-542\qtbase
Install prefix..............C:\Riverware\tools\Qt-542\qtbase
Headers installed to........C:\Riverware\tools\Qt-542\qtbase\include
Libraries installed to......C:\Riverware\tools\Qt-542\qtbase\lib
Arch-dep. data to...........C:\Riverware\tools\Qt-542\qtbase
Plugins installed to........C:\Riverware\tools\Qt-542\qtbase\plugins
Library execs installed to..C:\Riverware\tools\Qt-542\qtbase\bin
QML1 imports installed to...C:\Riverware\tools\Qt-542\qtbase\imports
QML2 imports installed to...C:\Riverware\tools\Qt-542\qtbase\qml
Binaries installed to.......C:\Riverware\tools\Qt-542\qtbase\bin
Arch-indep. data to.........C:\Riverware\tools\Qt-542\qtbase
Docs installed to...........C:\Riverware\tools\Qt-542\qtbase\doc
Translations installed to...C:\Riverware\tools\Qt-542\qtbase\translations
Examples installed to.......C:\Riverware\tools\Qt-542\qtbase\examples
Tests installed to..........C:\Riverware\tools\Qt-542\qtbase\tests
```

```
Info: creating super cache file C:\Riverware\tools\Qt-542\.qmake.super

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

C:\Riverware\tools\Qt-542>
```

(III.F) Building Qt5

Note: **It's best to capture and save** to a text file the content of the console having the Qt build configuration report (*as shown above*).

At this point ... just type nmake. The 64-bit builds -- *both debug and release* -- take about five (5) hours to complete.

As noted above "nmake confclean" doesn't actually work for preparing to re-run the configure script; that is apparently left over from Qt4.

Here are some metrics from this 64-bit build, compared to a similar build (also with Qt WebKit) for Qt 4.8.5:

C:\Riverware\tools\Qt-485	8.99 GB (9,657,989,499 bytes)	63,184 Files	7,380 Folders
C:\Riverware\tools\Qt-542	14.2 GB (15,348,694,767 bytes)	156,602 Files	19,682 Folders

(IV) RiverWare Porting Notes

Experimental RiverWare / Microsoft Visual Studio 2010 make files were created to test the Qt5 build. Adjustments to RiverWare source code were not made, so, of course, there are many compilation errors. Considering though, this went very well. By far, most of the required changes will be very simple. The make file changes are summarized below. This was also put away on a temporary **"RwQt5"** Git branch (*see the topmost commit*):

- https://cadswes2.colorado.edu/internal/cgi-bin/gitweb/gitweb.pl/builds.git/shortlog/refs/heads/RwQt5

RiverWare Build Changes: **Makefiles/qmake.pl:**

Old	`my $qt4 = 'C:\\RiverWare\\tools\\Qt-485';`
New	`my $qt4 = 'C:\\RiverWare\\tools\\Qt-542';`
Old	`my $cmd = "qmake $pro";`
New	`my $cmd = "C:/Riverware/tools/Qt-542/qtbase/bin/qmake $pro";`

RiverWare Build Changes: **Makefiles/riverwarebase.pro:**

Old	`QT += webkit`

New	`greaterThan(QT_MAJOR_VERSION, 4): QT += widgets`
	`lessThan(QT_MAJOR_VERSION, 5): QT += webkit`
	`greaterThan(QT_MAJOR_VERSION, 4): QT += webkitwidgets`
	`greaterThan(QT_MAJOR_VERSION, 4): QT += printsupport`

Here are some representative source code problems for a Qt5 build of RiverWare -- (all of this seems relatively minor).

1. Upper case boolean constants TRUE and FALSE are no longer defined.
2. QString::toAscii() no longer exists. Use QString::toLatin1() instead.
3. Less common stuff:
    1. QHeaderView::setResizeMode() no longer exists.
    2. Something about QtConcurrrentRun().
    3. No "+" operator for Qt::Modifier.
    4. Button widgets: setShown() no longer exists. Use setVisible() instead.

See the Qt5 Porting Guide and its linked topics [http://doc.qt.io/qt-5/portingguide.html] -- AND Transition from Qt 4.x to Qt5 [https://wiki.qt.io/Transition_from_Qt_4.x_to_Qt5].

Building **Qwt** (our Qt plotting package) with Qt5 is a significant effort. We are currently using Qwt 5.2.3. Porting this to Qt5 ourselves and ensuring that it is working properly may not be easy, and the result isn't ideal. We would need to maintain our "customized" (ported) Qwt code in a source code repository (e.g. Git). It would be better for us to upgrade to Qwt 6.1 -- that's the first version of Qwt intended to be used with Qt5. While this is not a minor upgrade, there are benefits. For example, Qwt 6.1 allows us to implement our own plot curve legends. Additional plot and scale types too.

--- (end) ---