Processing RPL Predefined Function Help for use in the RiverWare Build

Phil Weinstein, CADSWES, March 2015.

This document describes the process of exporting HTML from a FrameMaker 12 RiverWare Online Help document and proceessing that content for integration into the RiverWare build.

- Document Home: R:\doc\onlineHelp\process\GenRplFuncContent.html
- History:
 - 3-17-2015: Revised overview; reference to *feature* document.
 - $\circ\,$ 3-16-2015: Submitted for review. (Developed for RW 6.7).

Overview

RiverWare 6.7 displays RPL Predefined Function Online Help content, as HTML, in a QWebView panel. The initial implementation displays this help content in three RPL dialogs. See this document:

• RPL Predefined Function Help content in RiverWare 6.7 / March 2015 R:\doc\RPL\EmbedRplDoc\2015\RplFuncHelpInRiverWare-March2015.docx R:\doc\RPL\EmbedRplDoc\2015\RplFuncHelpInRiverWare-March2015.pdf

The Function HTML help content (a single HTML file for all, approximately 200 RPL Predefined Functions) is bound to the RiverWare executable as a Qt Resource. (A compressed copy of the page -- without images -- is part of the RiverWare executable). Referenced images are loaded dynamically by RiverWare's integrated QWebKit browser (Qt4 QWebView).

Major Process Steps:

(1) The RPL Predefined Function Online Help document is generated as HTML from FrameMaker 12. This is display-only oriented HTML content, lacking well-defined meta-data, e.g. for discerning which top-level HTML elements correspond to particular functions.

(2) A Python 3 implemented processor translates the FrameMaker HTML output into structured HTML, with semantic HTML/CSS classes and other XML attributes to identify functions (as new top-level HTML DIV elements) and the content components which make up a function block. The script also

 Controllegie
 Controllegie
 Controllegie
 Controllegie

 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controllegie
 Controll

- 🗆 ×

creates renamed copies of the exported images, with names based on the function blocks in which they appear. The script generates a directory containing:

- 1. A single HTML file with all RPL Predefined Function help content.
- 2. A subdirectory (folder) containing well-named images on that page.

(3) Automated: At run time, RiverWare builds an internal XML DOM document (in memory) from the bound HTML data and creates a function-name index for top-level function DIV elements. On demand, individual single function documents are built from those DIV elements and displayed in a Qt4 QWebKit QWebView. As mentioned above, this QWebView is deployed in three RPL Predefined Function-related dialogs.

Required Tools:

- FrameMaker 12 on Windows.
- Python 3. I'm using Python 3.4.0 (March 2014), 32-bit on my 64-bit Windows 7 system.
- A Windows RiverWare development area must be checked out from git (builds.git).

General Notes:

- The instructions below are for a RiverWare development area at "C:\Riverware\staff\philw \WinPhilDoc\". Adjust your process accordingly.
- The python script used in processing the HTML file is part of the RiverWare git repository (builds.git). Also checked in to that source repository are the direct input files and output files for that python script. This does *not* include the original FrameMaker source document for RPL Predefined Functions.
- The original FrameMaker source document lives in our central RiverWare/Doc area. It is under RCS source control, locally within its contained directory.

(1) If source document changes are needed ... check out <u>RPLPredefinedFunctions.fm</u> with RCS ... edit ... review ... and (*perhaps later on*) check in.

1. On "alamosa" (a linux machine), in a command window, check out the source document (with a lock) using RCS.

```
o cd /projects/riverware/doc/onlineHelp/builds/RPLPredefinedFunctions/
o co -l RPLPredefinedFunctions.fm
```

2. On Windows, START FrameMaker 12 and OPEN the document at the following analogous file path. (It's apparently normal to ignore the warnings about unavailable fonts and unresolved cross-references).

• R:\doc\onlineHelp\builds\RPLPredefinedFunctions\RPLPredefinedFunctions.fm

- 3. Carefully and deliberately edit the document using the style and format conventions of the document. Most styles are important -- they are used by the python script to identify the content structure and semantic types. Also important are the row title words in the first column of each function's main table -- those are also used as keywords by the python script, e.g. Description, Type, Arguments, [Argument Numbers, starting at 1], Evaluation and optional Comments. Math formula bounding boxes should not have a lot of extra internal padding space, especially on the left side.
- 4. Checking the edited documented back in (to RCS) can be done at any point in the following process. (It does not have to happen right now). But when you are pretty sure that your edits are complete and final, type in the following command in your "alamosa" (linux machine) command window to check your changes into RCS and unlock the file. (Refer to other RCS commands if, for some reason, the file you edited hadn't actually been locked).
 - o cd /projects/riverware/doc/onlineHelp/builds/RPLPredefinedFunctions/
 - ci -u RPLPredefinedFunctions.fm
 - ... type in a descriptive "change" message.

(2) Open <u>RPLPredefinedFunctions.fm</u> (again) in FrameMaker 12.

(3) File >> Publish ... shows this panel:

Publish	*≣
Current 🔽 🙆 🔁	Default 🖳
😈 Responsive HTML5	
🕜 Web Help	
ePub	
kindle	
🕲 Microsoft HTML Help	
I	
Output: C:\Users\philw\FMOutput	<u> </u>

(4) Confirm "US_ASCII" character encoding ...

- See screenshots below.
- Note: The Publish Settings dialog forces you to save the configuration change to a special file in order to apply it. Go ahead and do that, in a reasonable user folder in which you have "write" permissions.
- Notice the Output directory setting. This was the default (for the "philw" user).

Publish	→
Current 🔽 💋	Settings 🖳
Besponsive HTML5	Generate
 ✓ ePub ✓ Kindle ✓ Microsoft HTML Help 	View Edit Settings Explore
Output: C:\Users\philw\FMOut	put
ublish Settings: default	×
Style Mapping Outputs	
Responsive HTML5 WebHelp	General Navigation Search
ePub Kindle	Title: WebHelp
Microsoft HTML Help	Favicon: 😻 🖆
	Language: English (US)
	Encoding: US_ASCII
	Override Style Import
	Use Lowercase File Names (Recommended for UNIX)
	Save Save As Save and Close Cancel

(5) Double-click "WebHelp" ... this takes a few minutes ...

Progress	
Converting Source Files	
Converting "RPLPredeffm"	
Cancel	
Publish Result	X
C:\Users\philw\FMOutput\RPLPredefinedFunctions-WebHelp\index.html	
Manuface	
view Log view Output Do	one

... click Done.

(6) View these two folders:

- A. C:\Users\philw\FMOutput\RPLPredefinedFunctions-WebHelp\
- 1. Create or Delete Contents of Folder: [B] fmFuncImages.
- 2. Copy [A] <u>RPLPredefinedFunctions.htm</u> to [B].
- 3. Copy [A] <u>RPLPredefinedFunctions*.jpg</u> and <u>.bmp</u> to Folder: [B] fmFuncImages.

Note: In the course of making a sequence of text-only edits, sub-step #3 above -- *copying images* -- is not always necessary.

(7) Open a Windows command window inC:\Riverware\staff\philw\WinPhilDoc\RwDoc\PythonDocUtils\

• See screenshot below ... This can be done from the context menu in the containing folder. (Your Windows configuration may require the <SHIFT> key to be held when right-clicking on a folder for the "Open command line window here" operation to appear in the context menu).

🕌 C:\Riverware\staff\p	hilw\WinPhilDoc\RwDoc			
🕞 ि → ВООТ	(C:) ▼ Riverware ▼ staff ▼ philw ▼ Wir	PhilDoc 🔻 Rv	vDoc 🔻	👻 🐼 Search Rw
File Edit View Tools	Help			
Organize 🔻 🗦 Open	Include in library \checkmark Share with \checkmark	Burn Ne	w folder	
<u> </u> Google Drive	Name *		Date modified	Туре
En Librarian	퉬 Debug		3/14/2015 9:59 PM	File folder
Documents	퉬 embeddedHelp		3/13/2015 7:21 PM	File folder
A Music	🕌 PythonDocUtils	Open	0/10/0015 7 00 DM	
Pictures	🌗 qt	Open in n	ew window	
Videos	퉬 Release	Open com	nmand window here	
	Make.qt	7-Zip		<u> →</u>
🖳 Computer	RplPredefFuncDocViewer.cpp	KDiff3		+
🏭 BOOT (C:)	RplPredefFuncDocViewer.hpp	Scan with	System Center Endpoin	t Protection
👝 DATA (E:)	RwDoc.pro	Git Extens	sions	+
🖵 philw (\\alamc	RwDoc.grc	Share wit	h	•
philw (\\alamc	RwDoc.qrc	Share wit	h	•

(8) Run the python script

- Type processRplRedefFuncs.py (or python python processRplRedefFuncs.py)
- ... this is the expected result:

C:\Windows\system32\cmd.exe	
Function 182 SupplyNamesFrom	
Function 183 SupplySlotsFrom	
Function 184 SupplyNamesFromIntra	
Function 185 SupplySlotsFromIntra	
Function 186 SupplyNamesTo	
Function 187 SupplySlotsTo	
Function 188 SupplyNamesToIntra	
Function 189 SupplySlotsToIntra	
Function 190 TableInterpolation	
Function 191 TableInterpolation3D	
Function 192 TableLookup	
Function 193 TargetHWGivenInflow	
Function 194 TargetSlopeHWGivenInflow	
Function 195 ToCelcius	
Function 196 VolumeToFlow	
Function 197 WaterOwners	
Function 198 WaterTypes	
Function 199 WeightedSum	
HTML Generation:	
RiverWare RPL Predefined Functions Generated Mar 14, 2015 [23:07]	
Html Input : Inputs/RPLPredefinedFunctions.htm	
Html Output :/embeddedHelp/RplFunctionDoc.html	
Image Dir :/embeddedHelp/rplFuncImages/	
C:\Riverware\staff\philw\WinPhilDoc\RwDoc\PythonDocUtils>	-

The output of the python script is used in *two* places:

- 1. The RiverWare build. Only the generated HTML file (and not the page's *images*) becomes part of the RiverWare build. (See Step 9).
- 2. The RiverWare runtime file distribution. A directory including the generated HTML file *and* a subdirectory with the page's images is copied to the RiverWare executable directory -- or to the directory pointed to by the optional RiverWare Home environment variable. (See Step 10).

Note that both the "RwDoc/PythonDocUtils/Inputs/" directory *and* the generated the "RwDoc/embeddedHelp/" directory are part of the RiverWare development area under git control. That is, they both get checked in when committing changes in git.

(9) Build RiverWare in Visual Studio

- The change to file RwDoc/embeddedHelp/RplFunctionDoc.html -- as a result of running the python script in the prior step -- will cause a rebuild of the RwDoc library. This will be quick if the build area had otherwise been up to date. In general, only these modules will be built (in addition to the RiverWare executable itself):
 - a. RwDoc.qrc (with RCC, the Qt Resource compiler)
 - b. qrc_RwDoc.cpp
 - c. RwDocGenerated.lib

(10) Copy the generated directory to the RiverWare executable directory -- and ultimately to the RiverWare release deployment area.

- Copy the "RwDoc/embeddedHelp" subdirectory (folder) to the directory containing the RiverWare executable.
- The RiverWare program looks for the "embeddedHelp" directory under the program directory AND IN the directory pointed to by the optional RiverWare Home environment variable. If not found, the following message (for example) is sent to RiverWare Diagnostics:

The Embedded RiverWare Help directory (embeddedHelp) was not found under the RiverWare executable directory (C:/Riverware/staff/philw/WinPhilDoc /EngrObjs/debug) nor under the application directory (C:\Program Files\CADSWES\RiverWare 6.6\) defined with the optional application environment variable (RIVERWARE_HOME_67).

• During development (including revisions to the RPL Predefined Functions document), this step isn't always necessary. The HTML file (itself) under the executable directory isn't actually used by the running RiverWare program. Only the webpage's images under that directory are used. Those will not typically change, and nothing terrible happens if a new image is missing.

(11) Test, and with Git Extensions: commit and push changes.

- We use the "Git Extensions" Windows GUI application to manage local changes in git.
- Note: Of all of our developers, only one uses our <u>update-id.sh</u> utility to timestamp files before committing. (This is to be run in a Git Bash command line window at the top of the development tree; it effects only "staged" files, which must be re-staged after running this script). It is NOT IMPORTANT to run this utility. (Note that this had been an automatic feature of CVS. Git does not inherently support this timestamp provision). Not doing so will leave the original "\$Id\$" comment in the generated HTML file (which at least does include the file name). That will show up as a change to that file (as a difference indicated in Git Extensions). Not a problem. [This shell script file currently resides at C:\Riverware\bin\update-id.sh. It uses the set-id.pl perl script and git command line tools].

---- (end) ----