RiverWare Qt4 Port Completion -- Status: July 2010

Author: Phil Weinstein, CADSWES
Edit: 7-7-2010 / Status: Ready for Review (1) / [Word Document Copy](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.docx)
Revision: 7-8-2010 (b): minor edits, and added images to sections [2.2](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p2p4) and [2.3](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p3).
Revision: 7-8-2010 (c): QtRpl/RplListView (a [Q3ListView](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p2p1)) has already been replaced.

## (0.1) Overview

The last few RiverWare releases (5.0, 5.1 and 5.2) have been built with Qt version 4 (4.3 or 4.4) with use of the **"Qt3 Compatibility Library".** Graphical user interface (GUI) modules in RiverWare are now implemented using a mix of Qt4 widgets and Qt3 widgets (adapted for use with Qt4). To complete the Qt4 port, we need to recode all uses of Qt3 widgets to use instead Qt4 widgets. Benefits of completing the Qt4 port include improved user interface capabilities, performance, and ease of maintenance. Also, it will be important for all Qt3 to be removed from RiverWare before we can move to Qt 5 further down the road. (Qt is currently at version 4.6.3, with 4.7 in "beta" release).

The major high-level tasks in the Qt4 port completion are:

1. Converting all "Qt Designer" built UI files from Qt3 to Qt4.
2. Replacing all uses of Qt3 classes (widgets, etc.) with Qt4 classes.
3. Removing all uses of Qt3-compatibility methods in Qt4 classes.

See also the [analysis done in January 2010](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p2.html): http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p2.html

## (0.2) Status Summary

This table summarizes the progress of the Qt3 to Qt4 port with respect to the high-level tasks identified in January 2010. See additional information in subsequent numbered sections.

|  |  |  |
| --- | --- | --- |
| **TASK** | **January2010(Remaining)**  | **July2010 (Remaining)**  |
| [(1)](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect1) | Converting "Qt Designer" built UI files from Qt3 to Qt4 | [**91** UI Files](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p2.html#conv) | **15** UI Files (all DMI). |
| [(2)](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2) | Replacing all uses of Qt3 classes (widgets, etc) with Qt4 classes |
| [(2.1)](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p1) |    Porting Q3MainWindows to Qt4 QMainWindows     (not an enumerated development step in the Jan doc). | **46** | **10** (all DMI). |
| [(2.2)](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p2) |    Recoding all uses of Qt3 List and Table widgets |

|  |  |
| --- | --- |
| [Q3Lists](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p2.html#Q3List):  | **69** |
| [Q3Tables](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p2.html#Q3Tab):  | **4** |

 |

|  |  |
| --- | --- |
| Q3Lists:  | **48** (see below). |
| Q3Tables:  | **4** (see below). |

 |
| [(2.3)](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p3) |    Porting the Accounting View from Q3Canvas to       [Qt4] QGrahpicsView and QGraphicsScene. | TO DO | TO DO |
| [(2.4)](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p4) |    Porting RplFrame (a Q3ScrollView with custom     painting) | TO DO | TO DO |
| [(2.5)](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p5) |    Recoding all uses of other Qt3 Compatibility Library classes. | TO DO | About half done, in the course of other porting work. |
| [(3)](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect3) | Removing all uses of Qt3-compatibility methods in Qt4 classes. | TO DO | TO DO |

## (1) Converting "Qt Designer" built UI files from Qt3 to Qt4

**Background**

**Qt Designer** is a GUI building program supporting the definition and lay out of widgets for Qt dialogs and other GUI components. Designer maintains widget layout definitions in **"UI" (.ui) files**. Qt3 and Qt4 UI files have incompatible formats, and Qt4 Designer cannot directly read Qt3 Designer UI files. However, Qt4 provides a **"uic3"** (UI compiler) tool which has these two capabilities:

1. "uic3" can be used as part of the automated "make" process to generate C++ code from Qt3 UI files (using the Qt3 Compatibility Library). This is currently done in the RiverWare "make".
2. "uic3" can be used to convert Qt3 UI files to Qt4 UI files.

The process of using uic3 to convert Qt3 UI files to Qt4 UI files requires quiet a bit of hand-recoding. The major change is that the generated C++ class is no longer a base class for the client dialog. Instead, it is a POD ("plain old data") class which builds widgets inside of the client dialog, and is integrated with the client dialog by delegation (or "aggregation" -- either via a C++ pointer or reference data member). Qt4 Designer doesn't support custom Qt slots. Those Qt slots -- and the signal connections made to them -- need to be written in our C++ code. And icons defined in Qt3 UI files need to be handled differently.

See also this Trolltech webpage:[**Porting .ui Files to Qt 4**](http://doc.trolltech.com/4.4/porting4-designer.html)  [http://doc.trolltech.com/4.4/porting4-designer.html]

**Current Status**

This conversion has been applied to all Qt3 UI files except for the DMI-related UI files listed below. These were not addressed because it is possible that many of these UI files will be completely replaced with new ones.

* 1. DbDmi/DatasetMgrDlgBase.ui
	2. DbDmi/DbDmiDlgBase.ui
	3. DbDmi/DbDmiEditDlgBase.ui
	4. DbDmi/DbDmiImpExpDlgBase.ui
	5. DbDmi/DssDatasetDlgBase.ui
	6. DbDmi/HdbDatasetDlgBase.ui
	7. DbDmi/HdbMetaDataDlgBase.ui
	8. DbDmi/HdbModelRunIdDlgBase.ui
	9. DbDmi/HdbModelRunIdEditDlgBase.ui
	10. DbDmi/LoginDlgBase.ui
	11. DbDmi/NameMapDlgBase.ui
	12. DbDmi/NameMapMgrDlgBase.ui
	13. QtDmi/DmiParamDlgBase.ui
	14. QtDmi/ListCellDlgBase.ui
	15. QtDmi/MinMaxCellDlgBase.ui

The Qt3 UI file porting status has been documented on [this webpage](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Qt3UiFiles.html): http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Qt3UiFiles.html (Completed UI files are indicated with an orange background). A handful of UI files had been found to be no longer used in the RiverWare build.

**Implementation Notes**

* The Qt4 UI files generated with the Qt (4.4.3) "uic3" program can be loaded into Qt4 Designer, but they are not identical to the UI files saved from that program. In fact, some components from the Qt3 UI files which were preserved by the "uic3" program are lost when new Qt4 UI file is "round tripped" through (loaded into, and resaved from) Qt4 Designer. When doing this conversion, the "intermediate" state of the new Qt4 UI files (not yet round-tripped through designer) have been checked in to CVS.
* Icons: We have not yet started to use Qt's "resource file" (.qrc) capability for integrating icons (from independent image files). This is required for Qt4 Designer's support for icons in UI files. Instead, to date, icons have been removed from the converted UI files, and are placed in widgets in C++ code.
* The limitation of the uic3 conversion (as described in the webpage cited above) with respect to Qt Signals and Slots is overstated. Some "connections" are lost -- but not all. It seems that connections from only custom signals (and not to custom slots) are lost. In many cases though, all signal/slot connections were moved to C++ code, for consistency.
* Integration with the generated widget class code was done by delegation (or "aggregation") through C++ references rather than through C++ pointers. (I much prefer especially the "constness" afforded by the former approach).
* Screenshots, and more conversion details of particular sets of Qt3 UI files are provided on these webpages:
	+ [QtAccounting Library Qt3 UI Files](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Accounting/) ... [Notes from first ported module](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Accounting/AcctSysCfg/2010mar09b.txt)
	+ [QtUtils Library Qt3 UI Files](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/QtUtils/)
	+ [QtRun Library Qt3 UI Files](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/QtRun/) ... [See 3-12-2010 Status Message](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Email-2010mar12.txt).
	+ [Six Slot-related dialogs in Q3GUI: Qt3 UI Files](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Q3GUI-SlotSupport/)
	+ [Misc Remaining Q3GUI: Qt3 UI Files](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Q3GUI-Misc/)
	+ [Plotting Dialogs / Q3GUI Library Qt3 UI Files](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Plotting/)
	+ [QtSCT Library Qt3 UI Files](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/QtSCT/)
	+ [QtRpl Library Qt3 UI Files](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/QtRpl/)
	+ [QtDmi Library Qt3 UI Files](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/QtDmi/)

## (2) Replacing all uses of Qt3 classes (widgets, etc) with Qt4 classes.

All uses of the Qt3 Compatibility Library in Qt 4 need to be removed. This applies to both converted UI files and to our own C++ code. In UI files, this can be done mostly in Qt4 Designer -- though it is sometimes convenient to (carefully) directly edit the UI files (which are XML). Conveniently, the names of all Qt3 classes (in the Qt3 Compatibility Library) start with "Q3", e.g. Q3PopupMenu or Q3ListView.

This work is discussed in the following sections:

(2.1) [Porting Q3MainWindows to Qt4 QMainWindows](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p1)
(2.2) [Recoding all uses of Qt3 List and Table widgets](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p2)
(2.3) [Porting the Accounting View from Q3Canvas to QGrahpicsView](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p3).
(2.4) [Porting RplFrame](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p3) (a Q3ScrollView with custom painting)
(2.5) [Recoding all uses of other Qt3 Compatibility Library classes](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/PortAssessment5p3.html#Sect2p5).

## (2.1) Porting Q3MainWindows to Qt4 QMainWindows

Qt Dialogs are generally implemented with a subclass of either QDialog or QMainWindow. QMainWindows can have menubars, toolbars, statusbars, and child "dock" windows. Only QMainWindows have distinct Qt3-compatibility implementation (Q3MainWindow). In doing this conversion, generally only Q3MainWindows having toolbars and child "dock" windows need special treatment. Generally, the replacement of Q3MainWindow to QMainWindow can be, and was done mechanically, with one exception:

In all cases of Qt4 QMainWindows, a handle to the menubar needed to be obtained directly from the QMainWindow, rather than by explicitly creating a new QMenuBar for the QMainWindow, as was done in Qt3:

OLD: \_menuBar = new QMenuBar (this, "\_menuBar");
NEW: \_menuBar = menuBar(); // QMainWindow

As was mentioned above, much of the replacement of Q3MainWindow to QMainWindow was done mechanically, and checked with compilation and execution testing. This was done with several other Qt3-compatibility widget classes. See notes on these webpages:

* [Removing certain Q3 widgets from C++ source files in the Q3GUI library](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/2010jan15/)
* [Removing certain Q3 widgets from C++ source files in the non-DMI Qt libraries](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/2010jan16/)

Additional (19) Q3MainWindows which had been defined in Qt3 UI files were converted to QMainWindows after their UI files had been converted to Qt4. See notes on this webpage:

* [Qt4 Port: Q3MainWindow conversion to Qt4 QMainWindow](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Q3MainWindow/)

**Current Status**

This conversion has been applied to all Qt3MainWindows except for those in the following DMI-related UI files. These were not addressed because it is possible that many of these UI files will be completely replaced with new ones.

* + 1. DbDmi/DatasetMgrDlgBase.ui
		2. DbDmi/DbDmiDlgBase.ui
		3. DbDmi/DbDmiEditDlgBase.ui
		4. DbDmi/DssDatasetDlgBase.ui
		5. DbDmi/HdbDatasetDlgBase.ui
		6. DbDmi/NameMapDlgBase.ui
		7. DbDmi/NameMapMgrDlgBase.ui
		8. QtDmi/DmiParamDlgBase.ui
		9. QtDmi/ListCellDlgBase.ui
		10. QtDmi/MinMaxCellDlgBase.ui

## (2.2) Recoding all uses of Qt3 List and Table widgets

List and table widgets are the most numerous complex GUI components in RiverWare. Each dialog's Lists (including TreeViews) and Tables often have interactions with many other GUI and non-GUI software components. This, and the profound change in Qt4's List and Table widgets -- with the introduction of the new "model/view framework" -- has proved to make conversion of RiverWare's lists and tables quite time consuming.

Except in two very special cases, all of RiverWare's Qt3 lists and tables were "item-based" implementations. This means that a C++ object was associated with each row in a list, or each cell in a table. Qt4 also provides "item-based" lists and tables, but they are intended for only simple uses and have limitations, such as: (1) no custom drawing in cells and (2) difficulty in allowing editing in only certain columns. (The latter is possible, but requires a complex implementation). Qt4 "prefers" an item-less implementation for lists and tables. In the context of converting lists and tables which had been devised using an "item-based" architecture, this is unfortunate. In our Qt3 implementations, the "item objects" had interacted not only with the containing GUI component (list or table), but also directly with the data objects which those items represented (e.g. RiverWare Slots, SimObjs, Accounts, Output Devices, etc.).

The following approaches have been used (or considered) to port individual **Q3ListViews** to Qt4 widgets:

1. **Item-based: QTreeWidget** instance or subclass, with a QTreeWidgetItem subclass. This approach was used in porting most of the Q3ListViews in the QtAccounting library -- see notes on [this webpage](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Q3ListViews/2010jun10.txt): http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Q3ListViews/2010jun10.txt .
2. **Item-based: RwQListView** (a QTreeWidget) instance or subclass, with QTreeWidgetItem subclass. This is similar to the prior approach for porting Q3ListViews, but provided a central place to encapsulate special code common to most or all uses. Encapsulated capabilities could include RiverWare-standard functional configurations, common special features (e.g. moving the selected list items up or down in the list), and software engineering work-arounds (e.g. for optimal geometry management and other limitations of the QTreeWidget class). (We decided against this approach because it threatened the continued uniqueness of each table or list implementation in RiverWare). Porting Q3ListViews with this class was documented on [this webpage](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/guide/Q3ListView/): http://cadswes2.colorado.edu/~philw/2010/Qt4Port/guide/Q3ListView/ .
3. **Item-less:** Use of a **QStandardModel** and **QStandardItems** was briefly explored. They turn out to be a mismatch for Q3ListViews, as the items correspond to "cells" rather than to rows.
4. **Item-less: QTreeView** instance with: (1) a **QAbstractItemModel** subclass, (2) if sorting is needed: a **QSortFilterProxyModel** instance or subclass (see note), (3) if special drawing or editing is needed: a **QAbstractItemDelegate** subclass. It was found that the sorting provided by QSortFilterProxyModel was unstable when sorting a column having non-unique values. So, instead of using a direct instance, a general subclass, **RwQSortFilterProxyModel** was implemented to "cascade" to subsequent visible columns when equal values are found.

Although this is the "proper" Qt4 approach to implementing lists (and tree views), it results in quite a bit more code, especially in cases where a plain Q3ListView instance (rather than a subclass) had been used. Every single list instance will require at least a new QAbstractItemModel subclass. This approach is documented on the following webpage, and was applied to the Q3ListViews in the two indicated dialogs:

	* [**Porting RiverWare's Q3ListViews to Qt4 QTreeViews**](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Q3ListViews/PortingGuide.html)  (see class diagram below)
	   http://cadswes2.colorado.edu/~philw/2010/Qt4Port/Q3ListViews/PortingGuide.html
	* Dialog: [Q3GUI] ConfigSlotsDlg / ConfigSlotsSlotListModel
	* Dialog: [Q3GUI] SnapshotMgrDlg.cpp / SnapMgrObjListView / SnapMgrObjListView\_ObjItemData.cpp / SnapMgrObjListView\_ObjItemModel.cpp / SnapMgrSlotListView / SnapMgrSlotListView\_SlotItemData.cpp / SnapMgrSlotListView\_SlotItemModel.cpp.

|  |  |  |
| --- | --- | --- |
|         |

|  |
| --- |
| http://cadswes2.colorado.edu/%7Ephilw/2010/Qt4Port/Q3ListViews/ClassDiagram1-680.gifhttp://cadswes2.colorado.edu/~philw/2010/Qt4Port/Q3ListViews/ClassDiagram1.gif   |

 |

See the discussion on porting the several **Q3Tables** in the corresponding subsection, below.

### (2.2.1) Remaining Q3ListViews in HPP/CPP files

|  |
| --- |
| DbDmi/DbDmiMgr.hpp: Q3ListView\* \_listview; Q3GUI/AcctOrderListView.hpp: class AcctOrderListView : public Q3ListViewQ3GUI/DiagCfgDlg.Panel.hpp: Q3ListView\* \_catListView; Q3GUI/DiagOutputListView.hpp: class DiagOutputListView : public Q3ListViewQ3GUI/DisplayGroupListView.hpp: class DisplayGroupListView : public Q3ListViewQ3GUI/DragDropSimListView.hpp: class DragDropSimListView : public Q3ListViewQ3GUI/GusListView.hpp: class GusListView : public Q3ListView Q3GUI/OpenObjectDlg.hpp: Q3ListView\* \_methodsListView;Q3GUI/OpenObjectDlg.hpp: Q3ListView\* \_accountsListView;Q3GUI/OpenObjectDlg.hpp: Q3ListView\* \_olamListView;Q3GUI/RestoreListState.hpp: Q3ListView\* \_listView;Q3GUI/ScenarioMgrDlg.hpp: Q3ListView\* \_slotList; Q3GUI/SlotQListView.hpp: class SlotQListView : public Q3ListViewQ3GUI/SubbasinMgrListView.hpp: class SubbasinMgrListView : public Q3ListViewQ3GUI/TimeStepSelListView.hpp: class TimeStepSelListView : public Q3ListViewQ3GUI/WorkspaceListView.hpp: class WorkspaceListView : public Q3ListView QtRpl/RplBlockSelectorDlg.hpp: Q3ListView\* \_setListView; QtRun/DispatchInfoListView.hpp: class DispatchInfoListView : public Q3ListViewQtRun/DmiListView.hpp: class DmiListView : public Q3ListViewQtRun/QtMultiRunEditDlg.hpp: class QtMultiRunEditDlg::IterListView : public Q3ListViewQtRun/RulesEffectsListView.hpp: class RulesEffectsListView : public Q3ListView QtUtils/GenListView.hpp: class GenListView : public Q3ListView |

### (2.2.2) Remaining Q3ListViews in UI files

|  |
| --- |
| Q3GUI/FileInfoWidgets.ui: ... Q3ListView ... \_saveListQ3GUI/LinkEditorWidgets.ui: ... Q3ListView ... \_srcListViewQ3GUI/LinkEditorWidgets.ui: ... Q3ListView ... \_destListViewQ3GUI/MarkerMgrWidgets.ui: ... Q3ListView ... \_listViewQ3GUI/OpenObjectWidgets.ui: ... Q3ListView ... \_methodsListViewQ3GUI/OpenObjectWidgets.ui: ... Q3ListView ... \_accountsListViewQ3GUI/OutputConfigWidgets.ui: ... Q3ListView ... \_listSlotsQ3GUI/PlotMembershipWidgets.ui: ... Q3ListView ... \_listViewQ3GUI/PropagateWidgets.ui: ... Q3ListView ... \_listviewQ3GUI/SlotListWidgets.ui: ... Q3ListView ... \_slotListView QtDmi/DmiEditorWidgets.ui: ... Q3ListView ... \_listviewQtDmi/DmiHandlerWidgets.ui: ... Q3ListView ... \_msgListViewQtDmi/DmiMgrWidgets.ui: ... Q3ListView ... \_listviewQtDmi/DmiParamWidgets.ui: ... Q3ListView ... \_listviewQtDmi/ListCellWidgets.ui: ... Q3ListView ... \_listview QtRpl/RplAnalysisWidgets.ui: ... Q3ListView ... \_groupsListViewQtRpl/RplAnalysisWidgets.ui: ... Q3ListView ... \_descListViewQtRpl/RplAnalysisWidgets.ui: ... Q3ListView ... \_ascListViewQtRpl/RplImportWidgets.ui: ... Q3ListView ... \_importListViewQtRpl/RplLayoutWidgets.ui: ... Q3ListView ... \_breaksListViewQtRpl/RplPaletteWidgets.ui: ... Q3ListView ... \_userDefinedListViewQtRpl/RplPaletteWidgets.ui: ... Q3ListView ... \_predefinedListViewQtRpl/RplSearchReplaceWidgets.ui: ... Q3ListView ... \_searchResultsListView QtRun/MultiRunControlWidgets.ui: ... Q3ListView ... \_configListViewQtRun/OptParamSubWidgets.ui: ... Q3ListView ... \_paramListView |

### (2.2.3) Remaining Q3Tables

There are far fewer Q3Tables in RiverWare than Q3ListViews (see prior section). The Q3Tables for the **Open Slot Dialogs** (for Series and Table Slots) and for the **SCT** have a common base class **(SlotDataQTable),** and are quite complicated. These should be ported to the model-based **QTableView** (not QTableWidget) with a **QAbstractTableModel** subclass (to support an item-less implementation) and **QItemDelegate** (to support custom cell drawing). See our implementation for the Qt4 Run Analysis Dialog "grid" table classes: **RunAnalGridQTableView, RunAnalGridQItemDelegate,** and **RunAnalGridQTableModel.**

### Q3Table Subclasses:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |
| --- |
| **Q3GUI:** |
|   StatTableSlotDlg.hpp | : StatTableSlotDlg::SlotTable  | : Q3Table  (with items) |
|   SlotDataQTable.hpp  | : SlotDataQTable | : Q3Table  (itemless) |
|   SlotQDlgTable.hpp | : SlotQDlgTable  | : SlotDataQTable |
|   SctQTable.hpp | : SctQTable  | : SlotDataQTable |

 |

### Q3Table Direct Instances:

|  |  |  |  |
| --- | --- | --- | --- |
|

|  |
| --- |
| **Q3GUI:** |
|   RowColLabelEditor.hpp: | Q3Table\*  \_labelTable; |

 |

## (2.3) Porting the Accounting View from Q3Canvas to Qt4 QGraphicsView

This has been done for the Simulation View (from which the new Geospatial View has been adapted). With that example, and being that Qt3's QCanvasItems are analogous to Qt4's QGraphicsItems, converting the Accounting View to Qt4 is conceptually a fairly straight port. Also, some design documentation is available for that effort. Two intermediate QGraphicsScene base classes used for the Simulation View would be directly used also for the Accounting View, so many interactive and display features are already in place. (See class diagram below -- Accounting View classes are in Red).

See documents:

* **"**[**RiverWare Workspace port to Qt4 Graphics View**](http://cadswes2.colorado.edu/~philw/2009/GraphicsViewWorkspace/doc/Qt4GraphicsViewWorkspace-2009jan20.pdf)**"** (January 2009). [/projects/riverware/doc/guiRework/workspace/Qt4GraphicsViewWorkspace.fm and .pdf].
* **"**[**RiverWare 5.1 Simulation Workspace Enhancements**](http://cadswes2.colorado.edu/~philw/2009/GraphicsViewWorkspace/EnhancementNotes/Rw51SimWorkspaceEnhance-Phil2009may19.pdf)**"** (May 2009). [/projects/riverware/doc/guiRework/workspace/Rw51SimWorkspaceEnhance.fm and .pdf].
* [Trolltech] **"**[**Porting to Graphics View**](http://doc.trolltech.com/4.4/graphicsview-porting.html)**"** [http://doc.trolltech.com/4.4/graphicsview-porting.html].

|  |  |
| --- | --- |
|

|  |
| --- |
| http://cadswes2.colorado.edu/%7Ephilw/2009/GfxViewWS/diagrams/CanvasAndSceneClasses-720.gifhttp://cadswes2.colorado.edu/~philw/2009/GfxViewWS/diagrams/CanvasAndSceneClasses-2200.gif   |

 |

## (2.4) Porting RplFrame (a Q3ScrollView with custom painting)

RplFrame supports the display and editing of RPL Expressions. Probably, it would be best to port RplFrame (a Q3ScrollView with custom painting) to its analogous Qt4 widget: ScrollArea, as the current application-level implementation for rendering, selecting, and editing nested expressions works well and is, I believe, good code. . We could however consider reimplementing RplFrame using Qt's "managed graphics" capabilities (i.e. the Qt4 "Graphics View" architecture).

## (2.5) ****Recoding all uses of other Qt3 Compatibility Library classes****

This applies to our own C++ code as well as the widgets originally in Qt3 UI files not ported automatically by the uic3 tool, notably Q3ButtonGroup, Q3GroupBox, Q3MainWindow and Q3WidgetStack. With the observation that the tools used to port to Qt4 with the Qt3 Compatibility Library (in the preparation of RiverWare 5.0) were overly conservative, some of the unconverted Qt3-compatibility widgets were replaced using an automated process -- for example, generally all Q3PopupMenus were successfully changed to (Qt4) QMenu without further modification. See notes on these webpages:

* [Removing certain Q3 widgets from C++ source files in the Q3GUI library](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/2010jan15/)
* [Removing certain Q3 widgets from C++ source files in the non-DMI Qt libraries](http://cadswes2.colorado.edu/~philw/2010/Qt4Port/2010jan16/)

An updated approximate count of "Q3" widgets in RiverWare code has not been done since January. At that time, we approximated 585 (as C++ class data members, about half from Qt3 UI files). I suspect that we've removed at least half of those.

Progress on the subsequent step will be made easier when all Qt3 compatibility classes (whose names start with "Q3") have been removed from **all C++ header files.** But the goal, of course, is to remove references to those classes in all RiverWare code.

## (3) Removing all uses of Qt3-compatibility methods in Qt4 classes.

Many of the **Qt4 classes** include also **"Qt3 support members"** (generally, C++ methods). (See, for example, the [Qt4 QMenu class](http://qt.nokia.com/doc/4.4/qmenu.html)). Many of these are overridden methods, e.g. constructors with many initialization parameters -- those have generally been removed from Qt4. Once the prior step of removing all Qt3 classes form RiverWare header files has been completed, it will be useful to enable or disable availability of the Qt3 Compatibility Library for individual source files -- instead of having that library enabled for the entire build. [This is controlled with the **QT3\_SUPPORT** symbol].

--- (end) ---