EPICS QT Framework

2.0.0

Generated by Doxygen 1.7.5

Tue Dec 4 2012 23:01:25

Contents

Chapter 1

QE framework - EPICS aware Qt Widgets and data access classes

- QE is a layered software framework for accessing EPICS data using Channel Access on a range of platforms.

- The QE framework provides object oriented C++ access to control systems using EPICS (Experimental Physics and Industrial Control System). It is based on Qt, a widely used cross-platform application development framework.

- GUI or console based applications can be written that use QE at several levels. QE includes Qt plugin libraries, EPICS aware widgets, data formatting classes, and classes for accessing raw EPICS data in a Qt friendly way.

- QE also includes an application - QEgui - for displaying forms produced by the Qt development tool 'Designer'. Using this application a complete EPICS GUI system can be generated without writing any code. A GUI system produced in this way can interact with existing EPICS display tools such as EDM.

- QE handles much of the complexities of Channel Access including initiating and managing a channel. Applications using QE can interact with Channel Access using Qt based classes and data types. Channel Access updates are delivered using Qt's signals and slots mechanism.

1.1 Documentation

Support documents can be found in the documentation section of the epicsqt sourceforge project. The framework download (available on the epicsqt sourceforge homepage) also includes this documentation as well as full Doxygen generated documentation of all the epicsqt classes and widgets.

1.2 License

epicsqt is distributed under the terms of the GNU General Public License.

1.3 Platforms

epicsqt might be usable in all environments where you find Qt. It is compatible with Qt >= 4.4.

1.4 Screenshots

- ASgui screen shots
- other applications using epicsqt widgets
- Qt Designer
- Qt Creator

Screenshots are only available in the HTML docs.

1.5 Downloads

Stable releases and development snapshots are available at the epicsqt project page.

For getting a development snapshot from the SVN repository:

```
svn svn co https://epicsqt.svn.sourceforge.net/svnroot/epicsqt epicsqt
```

1.6 Installation

Read QE_GettingStarted.pdf in the documentation for setting up an enviroment for building or using the epicsqt framework.

To build the framework, open epicsqt.pro in QtCreator, ensure shaddow build is turned off, and hit build.

The resultant library libQEPlugin.so will need to be installed or referenced up according to how it is to be used - see QE_GettingStarted.pdf for details.

Any Qt specific queries? start at the Qt Project

1.7 Support

Visit the sourceforge epicsqt support page for assistance.

1.8 Related Projects

Qwt, The core of a Channel Access aware plotting widget.

1.9 Credits:

Authors:

Andrew Rhyder, Anthony Owen, Glenn Jackson

Project admin:

Andrew Rhyder <andrew.rhyder@synchrotron.org.au>

Chapter 2

GNU General Public License

The EPICS QT Framework is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The EPICS QT Framework is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the EPICS QT Framework.

If not, see "http://www.gnu.org/licenses/

Chapter 3

ASgui screen shots

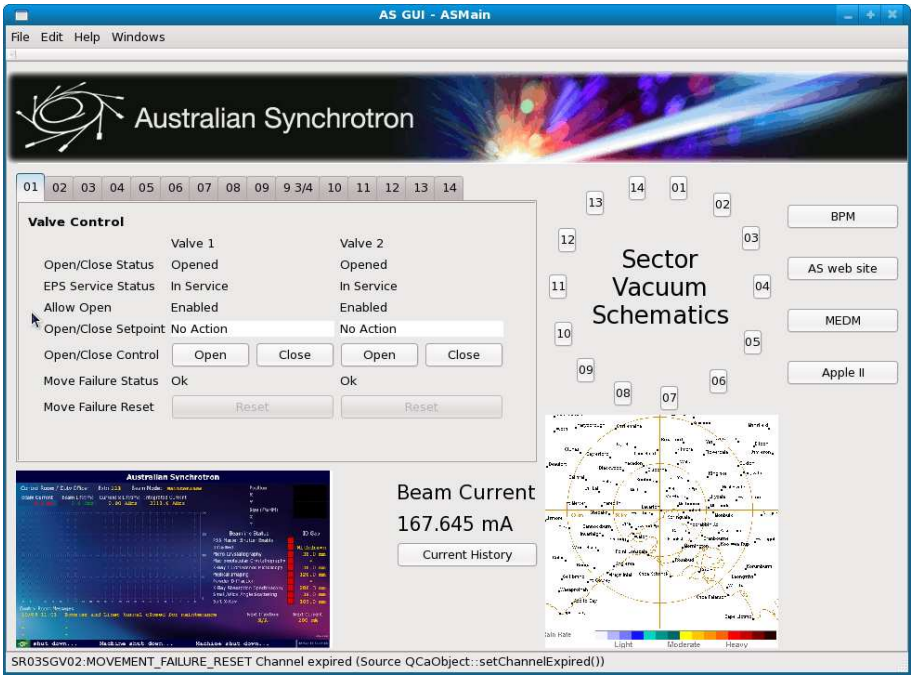


Figure 3.1: Australian Synchrotron mock up

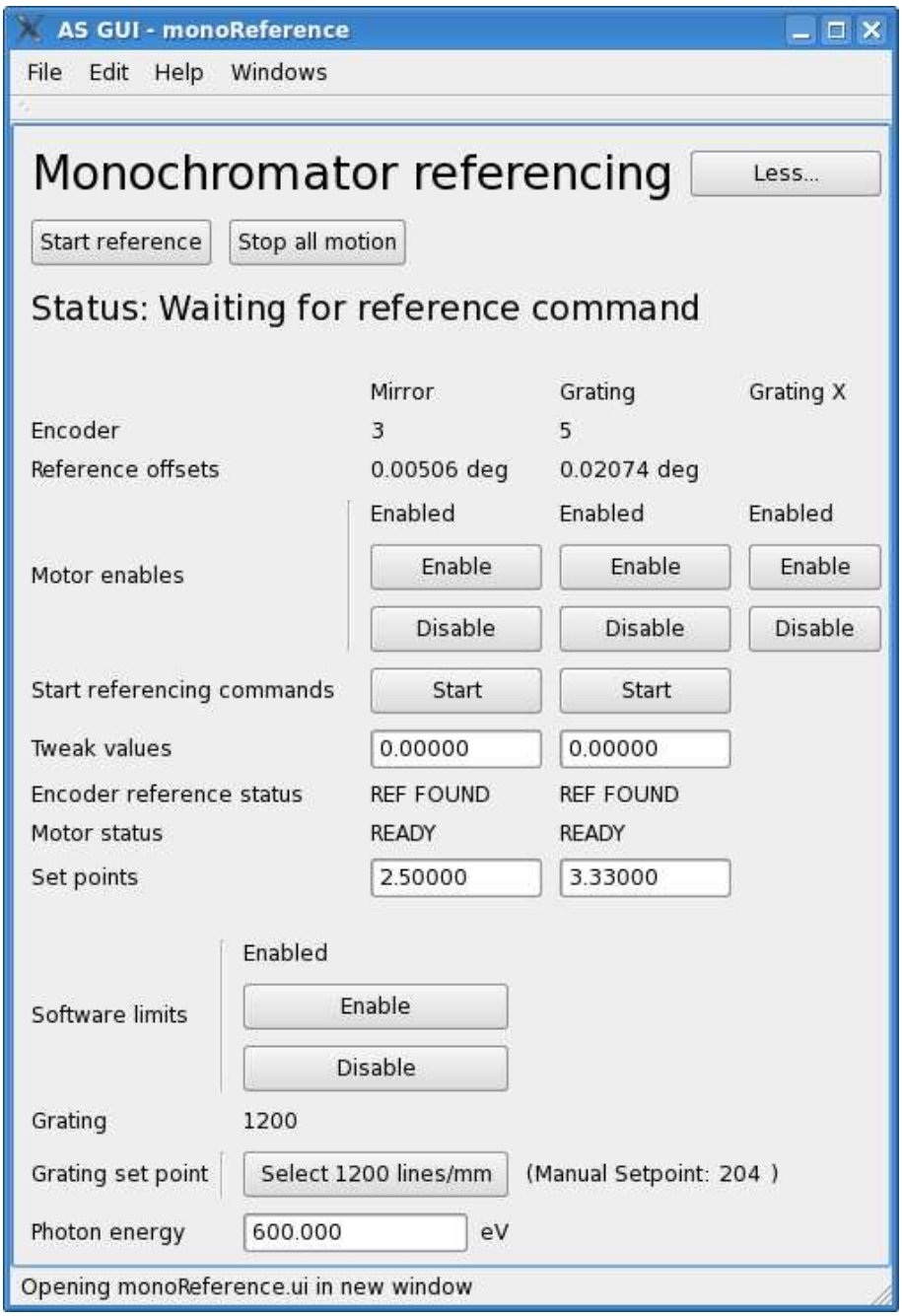


Figure 3.2: Monochromator referencing

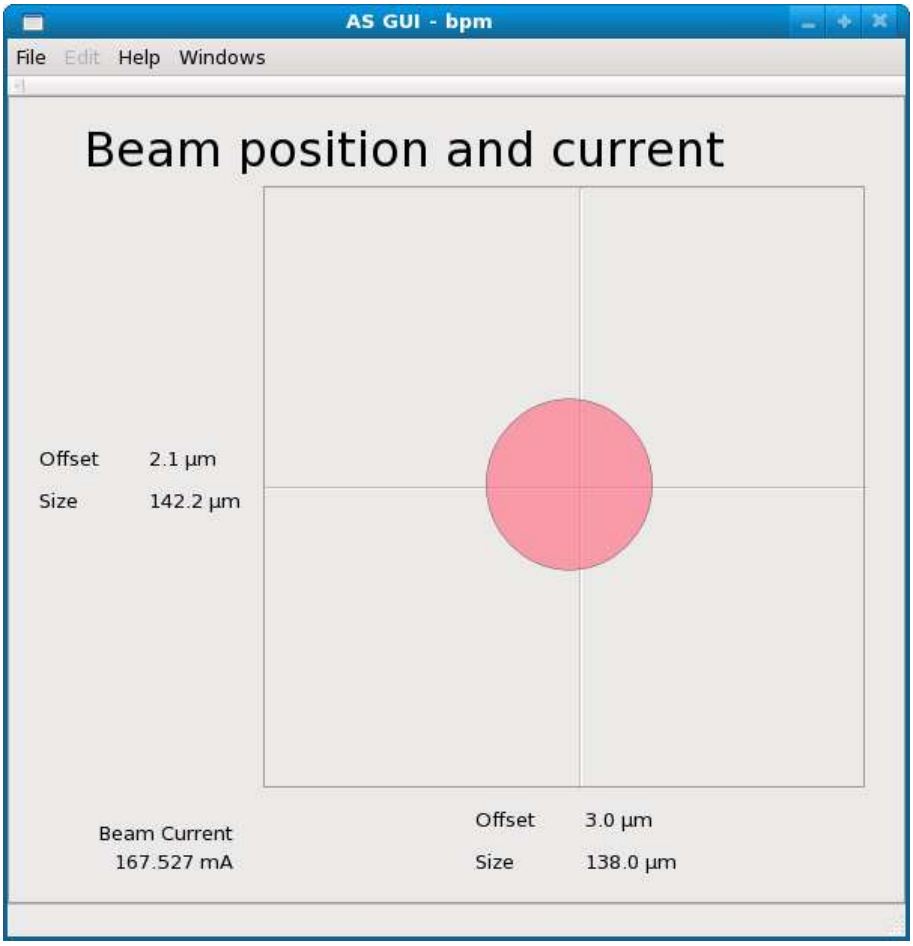


Figure 3.3: Beam position monitor

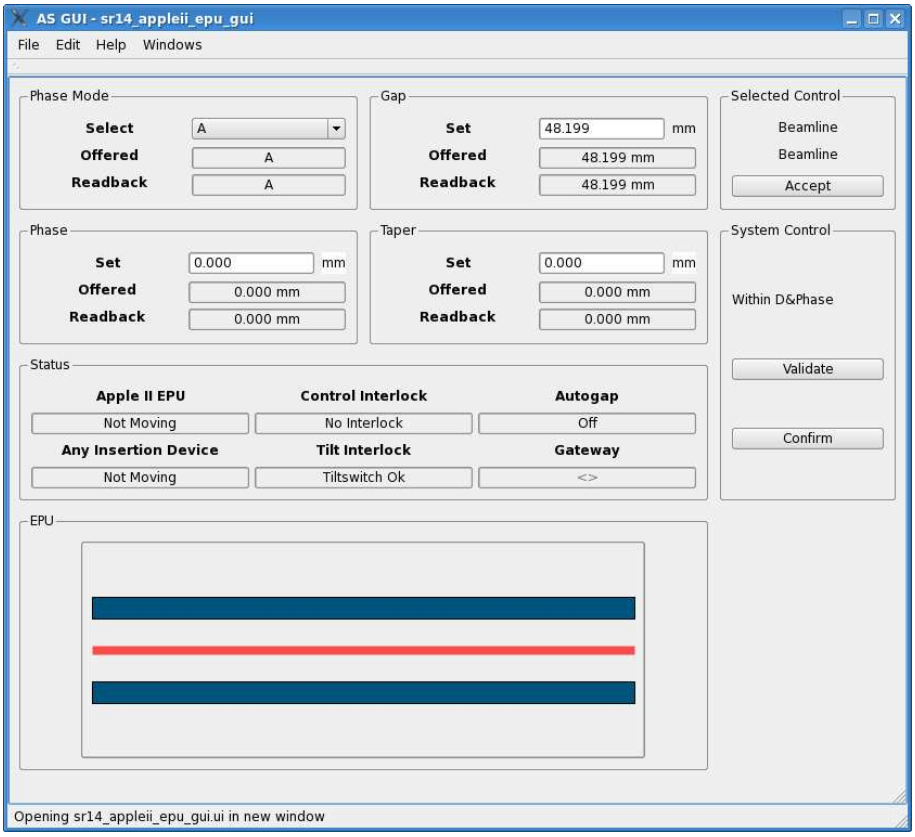


Figure 3.4: Insertion device

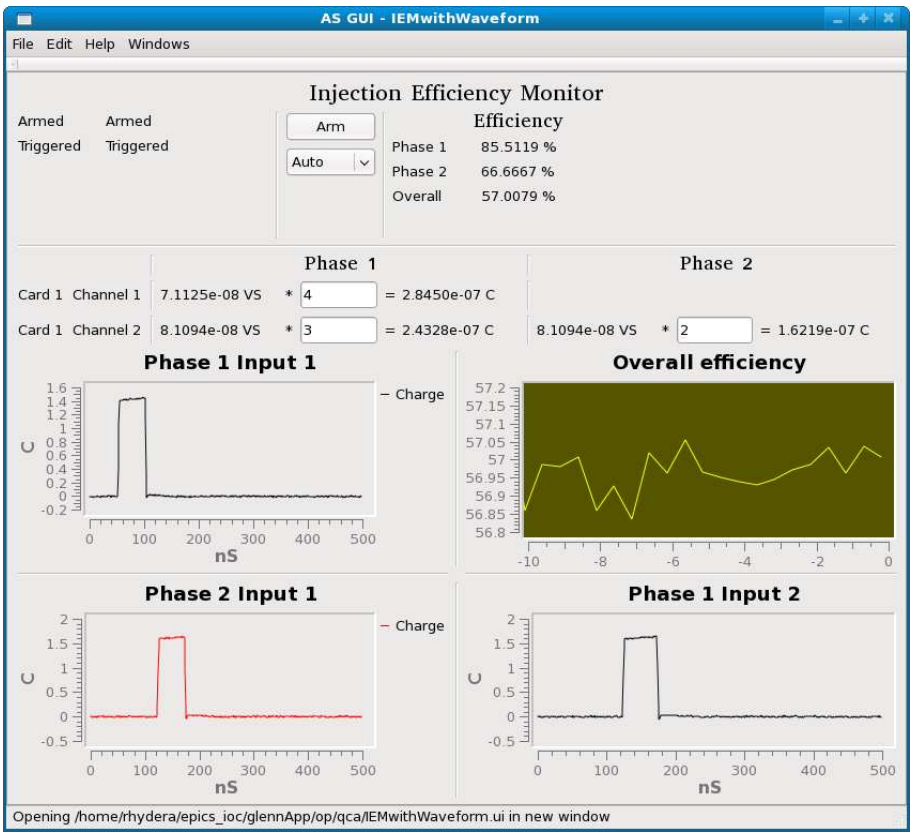


Figure 3.5: Injection efficiency monitor

Chapter 4

other applications using epicsqt widgets

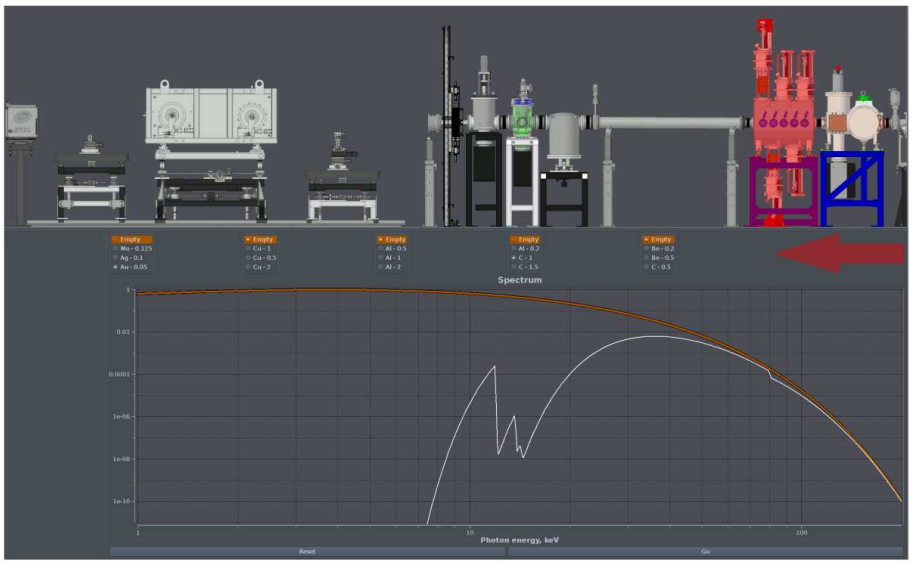


Figure 4.1: Medical Imaging beamline

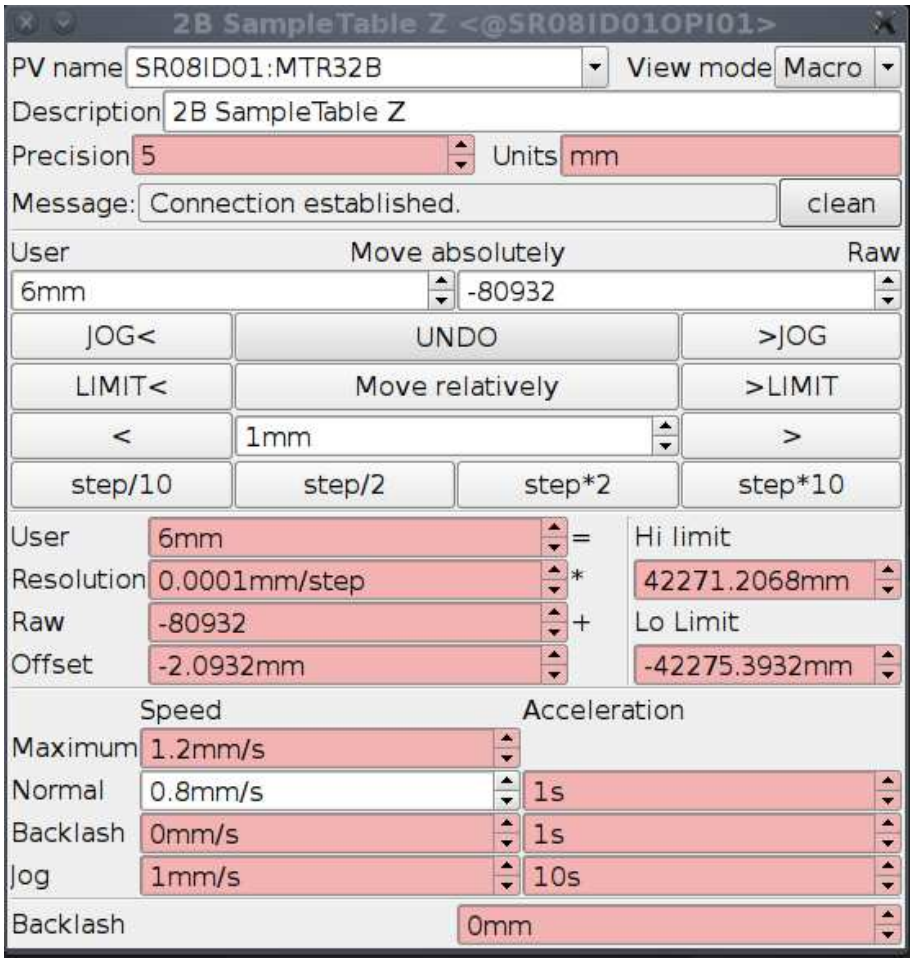


Figure 4.2: Motor controller

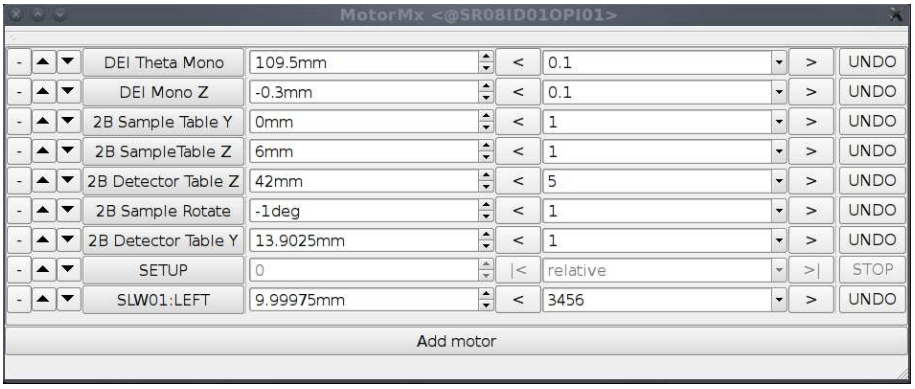


Figure 4.3: Motor controller

Chapter 5

Qt Designer

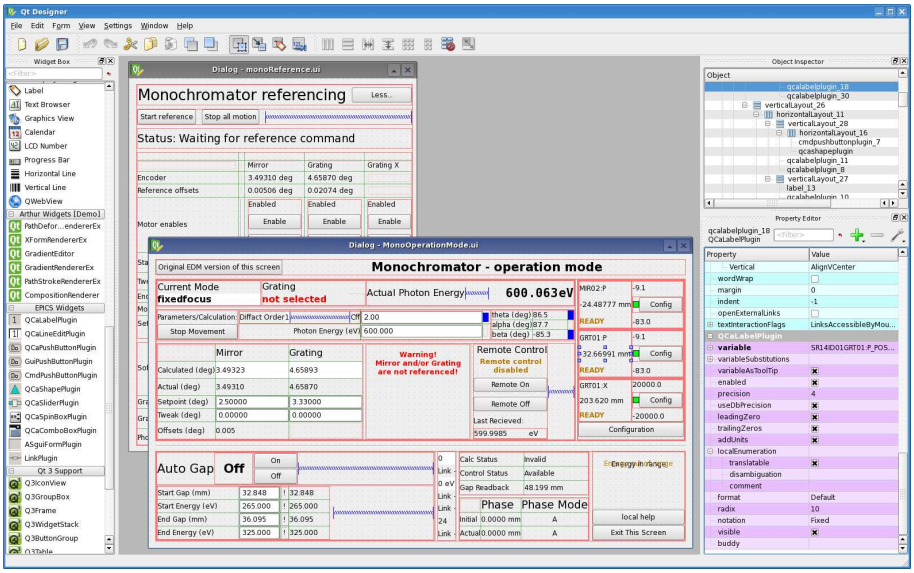


Figure 5.1: Editing multiple GUIs

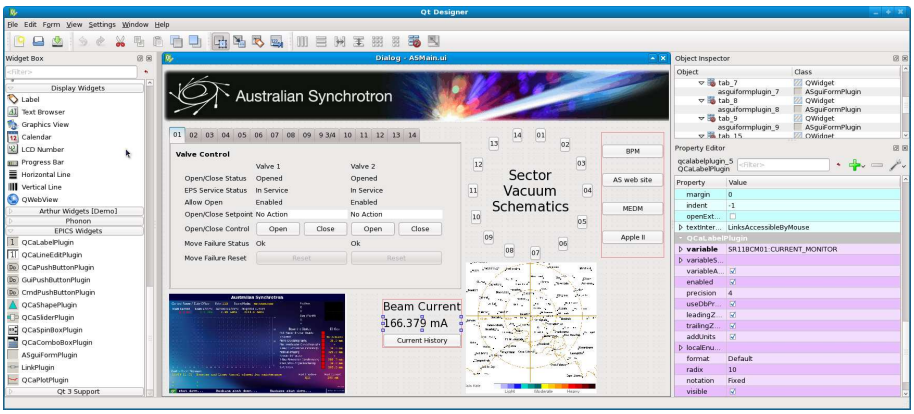


Figure 5.2: Editing a GUI

Chapter 6

Qt Creator

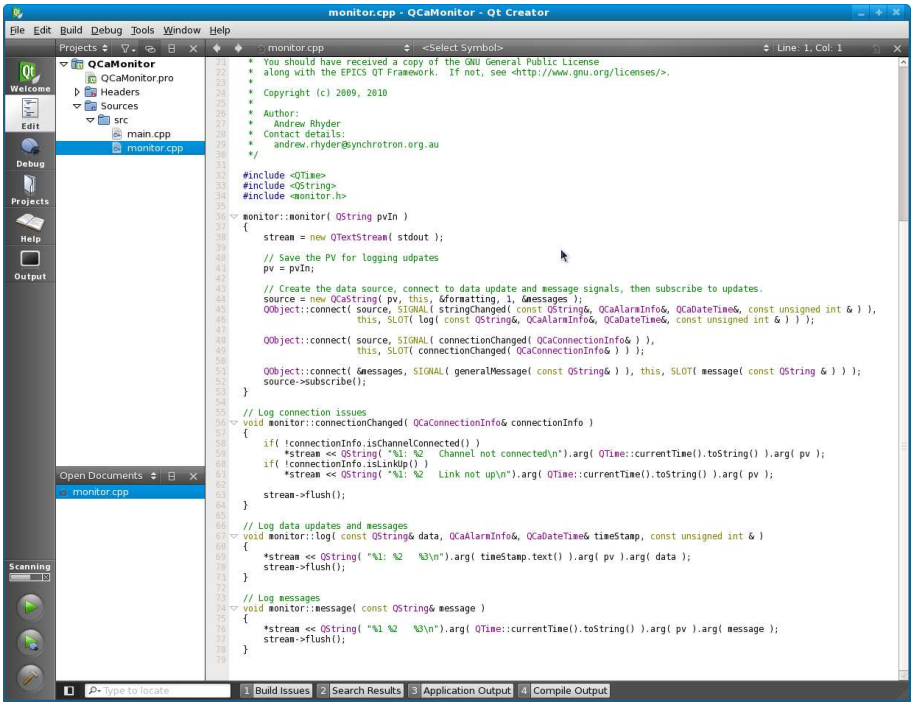


Figure 6.1: Application using epicsqt data source classes

Chapter 7

Class Index

7.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 8

Class Index

8.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 9

Class Documentation

9.1 _Field Class Reference

Public Member Functions

- QEWidget * **getWidget** ()
- void **setWidget** (QString *pValue)
- QString **getName** ()
- void **setName** (QString pValue)
- QString **getProcessVariable** ()
- void **setProcessVariable** (QString pValue)
- void **setJoin** (bool pValue)
- bool **getJoin** ()
- int **getType** ()
- void **setType** (int pValue)
- QString **getGroup** ()
- void **setGroup** (QString pValue)
- QString **getVisible** ()
- void **setVisible** (QString pValue)
- QString **getEditable** ()
- void **setEditable** (QString pValue)
- bool **getVisibility** ()
- void **setVisibility** (bool pValue)

Public Attributes

- QEWidget * **qCaWidget**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEConfiguredLayout/QEConfigured-Layout.h

- /home/andrew/epicsqt/framework/widgets/QEConfiguredLayout/QEConfigured-Layout.cpp

9.2 _Item Class Reference

Public Member Functions

- void **setName** (QString pValue)
- QString **getName** ()
- void **setSubstitution** (QString pValue)
- QString **getSubstitution** ()
- void **setVisible** (QString pValue)
- QString **getVisible** ()

Public Attributes

- QList< _Field * > **fieldList**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEConfiguredLayout/QEConfigured-Layout.h
- /home/andrew/epicsqt/framework/widgets/QEConfiguredLayout/QEConfigured-Layout.cpp

9.3 _QDialogItem Class Reference

Public Member Functions

- **_QDialogItem** (QWidget *pParent=0, QString pItemName="", QString pGroup-Name="", QList< _Field * > *pCurrentFieldList=0, Qt::WindowFlags pF=0)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEConfiguredLayout/QEConfigured-Layout.h
- /home/andrew/epicsqt/framework/widgets/QEConfiguredLayout/QEConfigured-Layout.cpp

9.4 _QDialogLogin Class Reference

Public Member Functions

- **_QDialogLogin** (QWidget *pParent=0, int pUserType=-1, Qt::WindowFlags pF=0)
- void **setCurrentUserType** (int pValue)
- void **setPassword** (QString pValue)

Protected Attributes

- QGridLayout * **qGridLayout**
- QVBoxLayout * **qVBoxLayout**
- QGroupBox * **qGroupBox**
- QRadioButton * **qRadioButtonUser**
- QRadioButton * **qRadioButtonScientist**
- QRadioButton * **qRadioButtonEngineer**
- QLabel * **qLabelType**
- QLineEdit * **qLineEditPassword**
- QPushButton * **qPushButtonOk**
- QPushButton * **qPushButtonCancel**
- int **userType**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QELogin/QELogin.h
- /home/andrew/epicsqt/framework/widgets/QELogin/QELogin.cpp

9.5 _QPushButtonGroup Class Reference

Public Slots

- void **buttonGroupClicked** ()

Public Member Functions

- **_QPushButtonGroup** (QWidget *pParent=0, QString pItemName="", QString pGroupName="", QList< _Field * > *pCurrentFieldList=0)
- void **mouseReleaseEvent** (QMouseEvent *qMouseEvent)
- void **keyPressEvent** (QKeyEvent *pKeyEvent)
- void **showDialogGroup** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEConfiguredLayout/QEConfigured-Layout.h
- /home/andrew/epicsqt/framework/widgets/QEConfiguredLayout/QEConfigured-Layout.cpp

9.6 _QTableWidgetFileBrowser Class Reference

Public Member Functions

- **_QTableWidgetFileBrowser** (QWidget ∗pParent=0)
- void **refreshSize** ()
- void **resizeEvent** (QResizeEvent ∗)
- void **resize** (int w, int h)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEFileBrowser/QEFileBrowser.h
- /home/andrew/epicsqt/framework/widgets/QEFileBrowser/QEFileBrowser.cpp

9.7 _QTableWidgetLog Class Reference

Public Member Functions

- **_QTableWidgetLog** (QWidget ∗pParent=0)
- void **refreshSize** ()
- void **resizeEvent** (QResizeEvent ∗)
- void **resize** (int w, int h)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QELog/QELog.h
- /home/andrew/epicsqt/framework/widgets/QELog/QELog.cpp

9.8 _QTableWidgetScript Class Reference

Public Member Functions

- **_QTableWidgetScript** (QWidget ∗pParent=0)
- void **refreshSize** ()
- void **resizeEvent** (QResizeEvent ∗)
- void **resize** (int w, int h)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEScript/QEScript.h
- /home/andrew/epicsqt/framework/widgets/QEScript/QEScript.cpp

9.9 QEAnalogIndicator::Band Struct Reference

Public Attributes

- double **lower**
- double **upper**
- QColor **colour**

The documentation for this struct was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEAnalogIndicator/QEAnalogIndicator.-h

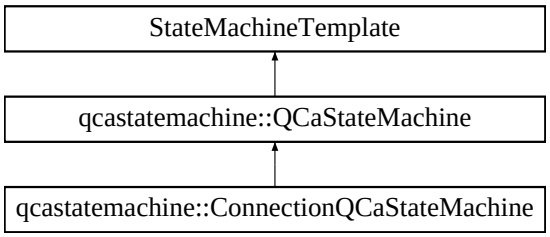
9.10 QEAnalogIndicator::BandList Class Reference

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEAnalogIndicator/QEAnalogIndicator.-h

9.11 qcastatemachine::ConnectionQCaStateMachine Class - Reference

Inheritance diagram for qcastatemachine::ConnectionQCaStateMachine:



Public Member Functions

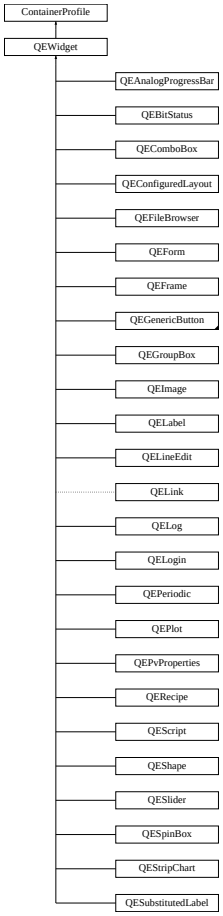
- **ConnectionQCaStateMachine** (void ∗parent)
- bool **process** (int requestedState)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QCaStateMachine.h
- /home/andrew/epicsqt/framework/data/src/QCaStateMachine.cpp

9.12 ContainerProfile Class Reference

Inheritance diagram for ContainerProfile:



Public Member Functions

- void **takeLocalCopy** ()
- void **setupProfile** (QObject ∗guiLaunchConsumerIn, QStringList pathListIn, QString parentPathIn, QString macroSubstitutionsIn)
- void **setupLocalProfile** (QObject ∗guiLaunchConsumerIn, QStringList pathListIn, QString parentPathIn, QString macroSubstitutionsIn)

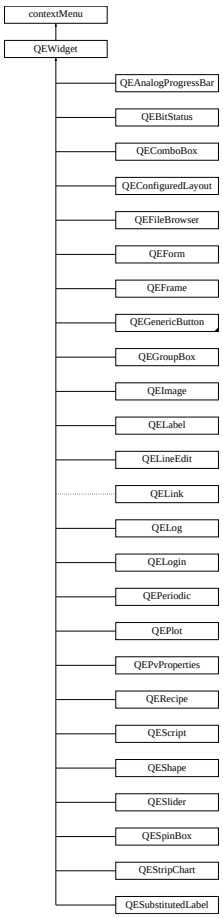
- void **updateConsumers** (QObject ∗guiLaunchConsumerIn)
- QObject ∗ **replaceGuiLaunchConsumer** (QObject ∗newGuiLaunchConsumerIn)
- void **addMacroSubstitutions** (QString macroSubstitutionsIn)
- void **removeMacroSubstitutions** ()
- QObject ∗ **getGuiLaunchConsumer** ()
- QString **getPath** ()
- QStringList **getPathList** ()
- QString **getParentPath** ()
- void **setPublishedParentPath** (QString publishedParentPathIn)
- QString **getMacroSubstitutions** ()
- bool **isProfileDefined** ()
- void **addContainedWidget** (QEWidget ∗containedWidget)
- QEWidget ∗ **getNextContainedWidget** ()
- void **removeContainedWidget** (QEWidget ∗containedWidget)
- unsigned int **getMessageFormId** ()
- unsigned int **getPublishedMessageFormId** ()
- void **setPublishedMessageFormId** (unsigned int publishedMessageFormIdIn)
- void **releaseProfile** ()
- void **publishOwnProfile** ()
- void **setUserLevel** (userLevels level)
- userLevels **getUserLevel** ()
- virtual void **userLevelChanged** (userLevels)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/ContainerProfile.h
- /home/andrew/epicsqt/framework/widgets/src/ContainerProfile.cpp

9.13 contextMenu Class Reference

Inheritance diagram for contextMenu:



Public Types

- enum **contextMenuOptions** { **CM_NONE**, **CM_COPY_VARIABLE**, **CM_COPY_DATA**, **CM_PASTE**, **CM_DRAG_VARIABLE**, **CM_DRAG_DATA**, **CM_SPECIFIC_WIDGETS_START_HERE** }

Public Member Functions

- void **addContextMenuToWidget** (QWidget ∗w)
- bool **isDraggingVariable** ()
- QMenu ∗ **getContextMenu** ()
- virtual QString **copyVariable** ()
- virtual QVariant **copyData** ()
- virtual void **paste** (QVariant)

Friends

- class **contextMenuObject**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/contextMenu.h
- /home/andrew/epicsqt/framework/widgets/src/contextMenu.cpp

9.14 contextMenuObject Class Reference

Public Slots

- void **contextMenuTriggered** (QAction ∗selectedItem)
- void **showContextMenu** (const QPoint &pos)
- void **setChecked** ()

Public Member Functions

- void **addContextMenuToWidget** (QWidget ∗w)
- void **manageChecked** (bool draggingVariable)
- void **setMenu** (contextMenu ∗menuIn)
- bool **isDraggingVariable** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/contextMenu.h
- /home/andrew/epicsqt/framework/widgets/src/contextMenu.cpp

9.15 QEPeriodic::elementInfoStruct Struct Reference

Public Attributes

- double **atomicWeight**
- QString **name**
- QString **symbol**
- double **meltingPoint**
- double **boilingPoint**
- double **density**
- unsigned int **group**
- double **ionizationEnergy**
- unsigned int **tableRow**
- unsigned int **tableCol**

Properties

- unsigned int **number**

The documentation for this struct was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEPeriodic/QEPeriodic.h

9.16 flipRotateMenu Class Reference

Public Member Functions

- **flipRotateMenu** (QWidget ∗parent=0)
- imageContextMenu::imageContextMenuOptions **getFlipRotate** (const QPoint &pos)
- void **setChecked** (const int rotation, const bool flipH, const bool flipV)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/flipRotateMenu.h
- /home/andrew/epicsqt/framework/widgets/QEImage/flipRotateMenu.cpp

9.17 imageContextMenu Class Reference

Public Types

- enum **imageContextMenuOptions** { **ICM_NONE** = contextMenu::CM_SPECIFIC_WIDGETS_START_HERE, **ICM_SAVE**, **ICM_PAUSE**, **ICM_ENABLE_TIME**, **ICM_ENABLE_CURSOR_PIXEL**, **ICM_ENABLE_CONTRAST_REVERSAL**, **ICM_ENABLE_PAN**, **ICM_ENABLE_VERT**, **ICM_ENABLE_HOZ**, **ICM_ENABLE_AREA**, **ICM_ENABLE_LINE**, **ICM_ENABLE_TARGET**, **ICM_DISPLAY_BUTTON_BAR**, **ICM_ZOOM_SELECTED**, **ICM_ZOOM_FIT**, **ICM_ZOOM_10**, × **ICM_ZOOM_25**, **ICM_ZOOM_50**, **ICM_ZOOM_75**, **ICM_ZOOM_100**, **ICM_ZOOM_150**, **ICM_ZOOM_200**, **ICM_ZOOM_300**, **ICM_ZOOM_400**, **ICM_ROTATE_NONE**, **ICM_ROTATE_RIGHT**, **ICM_ROTATE_LEFT**, **ICM_ROTATE_180**, **ICM_FLIP_HORIZONTAL**, **ICM_FLIP_VERTICAL**, **ICM_SELECT_PAN**, **ICM_SELECT_HSLICE**, **ICM_SELECT_VSLICE**, **ICM_SELECT_AREA**, **ICM_SELECT_PROFILE**, **ICM_SELECT_TARGET**, **ICM_SELECT_BEAM** }

Public Member Functions

- **imageContextMenu** (QWidget ∗parent=0)
- void **getContextMenuOption** (const QPoint &, imageContextMenuOptions ∗option, bool ∗checked)

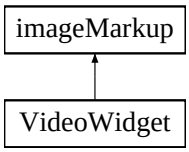
- void **addMenuItem** (const QString &title, const bool checkable, const bool checked, const imageContextMenuOptions option)
- void **addOptionMenuItem** (const QString &title, const bool checkable, const bool checked, const imageContextMenuOptions option)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/imageContextMenu.h
- /home/andrew/epicsqt/framework/widgets/QEImage/imageContextMenu.cpp

9.18 imageMarkup Class Reference

Inheritance diagram for imageMarkup:



Public Types

- enum **markupIds** { **MARKUP_ID_REGION**, **MARKUP_ID_H_SLICE**, **MARKUP_ID_V_SLICE**, **MARKUP_ID_LINE**, **MARKUP_ID_TARGET**, **MARKUP_ID_BEAM**, **MARKUP_ID_TIMESTAMP**, **MARKUP_ID_COUNT**, **MARKUP_ID_NONE** }

Public Member Functions

- void **markupMousePressEvent** (QMouseEvent ∗event)
- void **markupMouseReleaseEvent** (QMouseEvent ∗event)
- void **markupMouseMoveEvent** (QMouseEvent ∗event)
- void **setShowTime** (bool visibleIn)
- bool **getShowTime** ()
- markupIds **getMode** ()
- void **setMode** (markupIds modeIn)
- QVector< QRect > & **getMarkupAreas** ()
- bool **anyVisibleMarkups** ()
- QCursor **getDefaultMarkupCursor** ()
- void **setMarkupTime** (QCaDateTime &time)
- void **setMarkupColor** (markupIds mode, QColor markupColorIn)
- QColor **getMarkupColor** (markupIds mode)
- QCursor **getCircleCursor** ()
- QCursor **getTargetCursor** ()
- virtual void **markupSetCursor** (QCursor cursor)=0

Public Attributes

- QImage * **markupImage**
- QVector< markupItem * > **items**
- QPoint **grabOffset**
- bool **markupAreasStale**

Protected Member Functions

- void **markupResize** (QSize newSize)
- virtual void **markupChange** (QImage &markups, QVector< QRect > &changedAreas)=0
- virtual void **markupAction** (markupIds mode, QPoint point1, QPoint point2)=0

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.h
- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.cpp

9.19 localEnumerationItem Class Reference

Public Types

- enum **operations** { **LESS**, **LESS_EQUAL**, **EQUAL**, **NOT_EQUAL**, **GREATER_EQUAL**, **GREATER**, **ALWAYS**, **UNKNOWN** }

Public Attributes

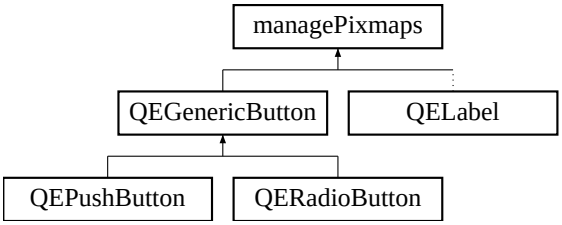
- double **dValue**
- QString **sValue**
- operations **op**
- QString **text**

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/data/include/QEStringFormatting.h

9.20 managePixmaps Class Reference

Inheritance diagram for managePixmaps:



Public Member Functions

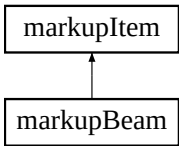
- void **setDataPixmap** (const QPixmap &Pixmap, const unsigned int index)
- QPixmap **getDataPixmap** (const unsigned int index)
- QPixmap **getDataPixmap** (const QString value)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/managePixmaps.h
- /home/andrew/epicsqt/framework/widgets/src/managePixmaps.cpp

9.21 markupBeam Class Reference

Inheritance diagram for markupBeam:



Public Member Functions

- **markupBeam** (imageMarkup ∗ownerIn, bool interactiveIn, bool reportOnMoveIn)

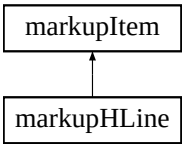
- void **startDrawing** (QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (QPoint pos)
- bool **isOver** (QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.h
- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.cpp

9.22 markupHLine Class Reference

Inheritance diagram for markupHLine:



Public Member Functions

- **markupHLine** (imageMarkup ∗ownerIn, bool interactiveIn, bool reportOnMoveIn)

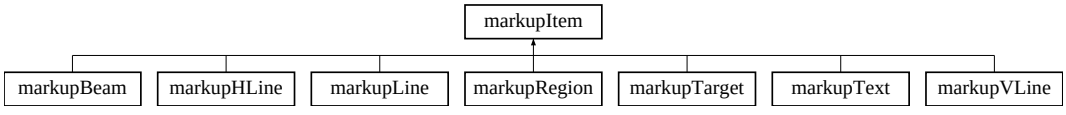
- void **startDrawing** (QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (QPoint pos)
- bool **isOver** (QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.h
- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.cpp

9.23 markupItem Class Reference

Inheritance diagram for markupItem:



Public Types

- enum **isOverOptions** { **OVER_LINE**, **OVER_BORDER**, **OVER_AREA** }
- enum **markupHandles** { **MARKUP_HANDLE_NONE**, **MARKUP_HANDLE_START**, **MARKUP_HANDLE_END**, **MARKUP_HANDLE_TL**, **MARKUP_HANDLE_TR**, **MARKUP_HANDLE_BL**, **MARKUP_HANDLE_BR**, **MARKUP_HANDLE_T**, **MARKUP_HANDLE_B**, **MARKUP_HANDLE_L**, **MARKUP_HANDLE_R** }

Public Member Functions

- **markupItem** (imageMarkup *ownerIn, isOverOptions over, bool interactiveIn, bool reportOnMoveIn)
- virtual void **setArea** ()=0
- virtual QPoint **origin** ()=0
- virtual void **moveTo** (QPoint pos)=0
- void **erase** ()
- virtual void **drawMarkup** (QPainter &p)=0
- virtual void **startDrawing** (QPoint pos)=0
- virtual bool **isOver** (QPoint point, QCursor *cursor)=0
- virtual QPoint **getPoint1** ()=0
- virtual QPoint **getPoint2** ()=0
- virtual QCursor **defaultCursor** ()=0
- bool **pointIsNear** (QPoint p1, QPoint p)
- void **drawMarkupIn** ()
- void **drawMarkupOut** ()
- void **setColor** (QColor colorIn)
- QColor **getColor** ()

Public Attributes

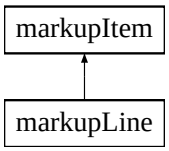
- markupHandles **activeHandle**
- isOverOptions **isOverType**
- QRect **area**
- bool **visible**
- bool **interactive**
- bool **reportOnMove**
- bool **highlighted**
- int **highlightMargin**
- QColor **color**
- imageMarkup * **owner**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.h
- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.cpp

9.24 markupLine Class Reference

Inheritance diagram for markupLine:



Public Member Functions

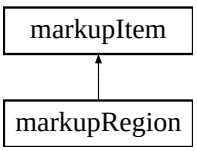
- **markupLine** (imageMarkup ∗ownerIn, bool interactiveIn, bool reportOnMoveIn)
- void **startDrawing** (QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (QPoint pos)
- bool **isOver** (QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.h
- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.cpp

9.25 markupRegion Class Reference

Inheritance diagram for markupRegion:



Public Member Functions

- **markupRegion** (imageMarkup ∗ownerIn, bool interactiveIn, bool reportOnMoveIn)
- void **startDrawing** (QPoint pos)

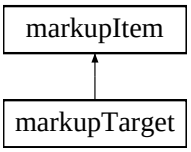
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (QPoint pos)
- bool **isOver** (QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.h
- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.cpp

9.26 markupTarget Class Reference

Inheritance diagram for markupTarget:



Public Member Functions

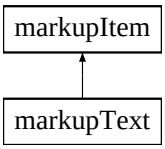
- **markupTarget** (imageMarkup ∗ownerIn, bool interactiveIn, bool reportOnMoveIn)
- void **startDrawing** (QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (QPoint pos)
- bool **isOver** (QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.h
- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.cpp

9.27 markupText Class Reference

Inheritance diagram for markupText:



Public Member Functions

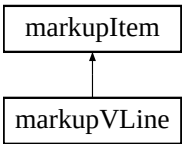
- **markupText** (imageMarkup ∗ownerIn, bool interactiveIn, bool reportOnMoveIn)
- void **setText** (QString textIn, bool draw)
- void **startDrawing** (QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (QPoint pos)
- bool **isOver** (QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.h
- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.cpp

9.28 markupVLine Class Reference

Inheritance diagram for markupVLine:



Public Member Functions

- **markupVLine** (imageMarkup ∗ownerIn, bool interactiveIn, bool reportOnMoveIn)

- void **startDrawing** (QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (QPoint pos)
- bool **isOver** (QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.h
- /home/andrew/epicsqt/framework/widgets/QEImage/imageMarkup.cpp

9.29 PeriodicDialog Class Reference

Public Member Functions

- **PeriodicDialog** (QWidget ∗parent=0)
- QString **getElement** ()
- void **setElement** (QString elementIn, QList< bool > &enabledList, QList< QString > &elementList)

Protected Member Functions

- void **changeEvent** (QEvent ∗e)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEPeriodic/PeriodicDialog.h
- /home/andrew/epicsqt/framework/widgets/QEPeriodic/PeriodicDialog.cpp

9.30 PeriodicElementSetupForm Class Reference

Public Member Functions

- **PeriodicElementSetupForm** (QWidget ∗parent=0)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEPeriodic/PeriodicElementSetupForm.h
- /home/andrew/epicsqt/framework/widgets/QEPeriodic/PeriodicElementSetupForm.cpp

9.31 PeriodicSetupDialog Class Reference

Public Member Functions

- **PeriodicSetupDialog** (QWidget *parent=0)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEPeriodic/PeriodicSetupDialog.h
- /home/andrew/epicsqt/framework/widgets/QEPeriodic/PeriodicSetupDialog.cpp

9.32 QEStripChart::PrivateData Class Reference

Public Member Functions

- **PrivateData** (QEStripChart *chartIn)
- QEStripChartItem * **getItem** (unsigned int slot)
- QwtPlotCurve * **allocateCurve** ()
- void **calcDisplayMinMax** ()
- void **plotData** ()
- void **setReadOut** (QString text)

Public Attributes

- enum ChartYScale **chartYScale**
- enum ChartTimeMode **chartTimeMode**

Protected Member Functions

- bool **eventFilter** (QObject *obj, QEvent *event)

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChart.cpp

9.33 QEStripChartItem::PrivateData Class Reference

Public Attributes

- QEStripChart * **chart**
- QLabel * **pvName**
- QELabel * **caLabel**

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChartItem.cpp

9.34 profilePlot Class Reference

Public Member Functions

- **profilePlot** (QWidget ∗parent=0)
- void **setScale** (int scaleIn)
- void **setProfile** (QVector< QPointF > &profile, double minX, double maxX, double minY, double maxY)
- void **setCursor** (int cursorIn)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/profilePlot.h
- /home/andrew/epicsqt/framework/widgets/QEImage/profilePlot.cpp

9.35 PushButtonSpecifications Struct Reference

Public Attributes

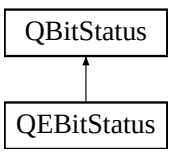
- int **width**
- const QString **caption**
- const QString **iconName**
- const QString **toolTip**
- const char ∗ **member**

The documentation for this struct was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChart.cpp

9.36 QBitStatus Class Reference

Inheritance diagram for QBitStatus:



Public Types

- enum **Orientations** { **LSB_On_Right**, **LSB_On_Bottom**, **LSB_On_Left**, **LSB_-On_Top** }
- enum **Shapes** { **Rectangle**, **Circle** }

Public Slots

- void **setValue** (const int value)

Public Member Functions

- **QBitStatus** (QWidget ∗parent=0)
- virtual QSize **sizeHint** () const
- void **setBorderColour** (const QColor value)
- QColor **getBorderColour** ()
- void **setOnColour** (const QColor value)
- QColor **getOnColour** ()
- void **setOffColour** (const QColor value)
- QColor **getOffColour** ()
- void **setInvalidColour** (const QColor value)
- QColor **getInvalidColour** ()
- void **setClearColour** (const QColor value)
- QColor **getClearColour** ()
- void **setDrawBorder** (const bool value)
- bool **getDrawBorder** ()
- void **setNumberOfBits** (const int value)
- int **getNumberOfBits** ()
- void **setGap** (const int value)
- int **getGap** ()
- void **setShift** (const int value)
- int **getShift** ()
- void **setOnClearMask** (const QString value)
- QString **getOnClearMask** ()
- void **setOffClearMask** (const QString value)
- QString **getOffClearMask** ()
- void **setReversePolarityMask** (const QString value)
- QString **getReversePolarityMask** ()
- void **setIsValid** (const bool value)
- bool **getIsValid** ()
- void **setOrientation** (const enum Orientations value)
- enum Orientations **getOrientation** ()
- void **setShape** (const enum Shapes value)
- enum Shapes **getShape** ()
- int **getValue** ()

Properties

- int **value**
- int **numberOfBits**
- int **shift**
- Orientations **Orientation**
- Shapes **shape**
- int **gap**
- QString **reversePolarityMask**
- QString **onClearMask**
- QString **offClearMask**
- QColor **boarderColour**
- QColor **invalidColour**
- QColor **onColour**
- QColor **offColour**
- QColor **clearColour**
- bool **drawBorder**
- bool **isValid**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEBitStatus/QBitStatus.h
- /home/andrew/epicsqt/framework/widgets/QEBitStatus/QBitStatus.cpp

9.37 QCaAlarmInfo Class Reference

Public Member Functions

- **QCaAlarmInfo** (unsigned short statusIn, unsigned short severityIn)
- QString **statusName** ()
- QString **severityName** ()
- bool **isInAlarm** ()
- bool **isMinor** ()
- bool **isMajor** ()
- bool **isInvalid** ()
- QString **style** ()
- QString **getColorName** ()
- QCAALARMINFO_SEVERITY **getSeverity** ()

Static Public Member Functions

- static QCAALARMINFO_SEVERITY **getInvalidSeverity** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QCaAlarmInfo.h
- /home/andrew/epicsqt/framework/data/src/QCaAlarmInfo.cpp

9.38 QCaConnectionInfo Class Reference

Public Member Functions

- **QCaConnectionInfo** (unsigned short channelStateIn, unsigned short linkStateIn)
- bool **isChannelConnected** ()
- bool **isLinkUp** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QCaConnectionInfo.h
- /home/andrew/epicsqt/framework/data/src/QCaConnectionInfo.cpp

9.39 QCaDataPoint Struct Reference

Public Attributes

- double **value**
- QCaDateTime **datetime**
- QCaAlarmInfo **alarm**

The documentation for this struct was generated from the following file:

- /home/andrew/epicsqt/framework/data/include/QCaDataPoint.h

9.40 QCaDataPointList Class Reference

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/data/include/QCaDataPoint.h

9.41 QCaDateTime Class Reference

Public Member Functions

- **QCaDateTime** (QDateTime dt)
- void **operator=** (const QCaDateTime &other)
- **QCaDateTime** (unsigned long seconds, unsigned long nanoseconds)
- QString **text** ()
- double **floating** (QDateTime base)

Public Attributes

- unsigned long **nSec**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QCaDateTime.h
- /home/andrew/epicsqt/framework/data/src/QCaDateTime.cpp

9.42 QCaEventFilter Class Reference

Public Member Functions

- void **addFilter** (QObject ∗objectIn)
- void **deleteFilter** (QObject ∗objectIn)
- bool **eventFilter** (QObject ∗watched, QEvent ∗e)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QCaEventFilter.h
- /home/andrew/epicsqt/framework/data/src/QCaEventFilter.cpp

9.43 QCaEventItem Class Reference

Public Member Functions

- **QCaEventItem** (QCaEventUpdate ∗newEvent)

Public Attributes

- QCaEventUpdate ∗ **event**

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/data/include/QCaEventUpdate.h

9.44 QCaEventUpdate Class Reference

Public Member Functions

- **QCaEventUpdate** (qcaobject::QCaObject ∗emitterObjectIn, long newReason, void ∗newDataPtr)

Public Attributes

- bool **acceptThisEvent**
- qcaobject::QCaObject * **emitterObject**
- long **reason**
- void * **dataPtr**

Static Public Attributes

- static QEvent::Type **EVENT_UPDATE_TYPE** = QEvent::User

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QCaEventUpdate.h
- /home/andrew/epicsqt/framework/data/src/QCaEventUpdate.cpp

9.45 QCaInstalledFiltersListItem Class Reference

Public Member Functions

- **QCaInstalledFiltersListItem** (QObject *eventObjectIn)

Public Attributes

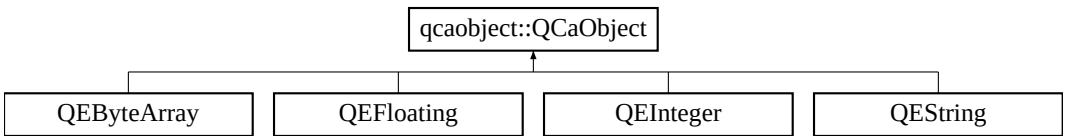
- QObject * **eventObject**
- long **referenceCount**

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/data/include/QCaEventFilter.h

9.46 qcaobject::QCaObject Class Reference

Inheritance diagram for qcaobject::QCaObject:



Public Slots

- bool **writeData** (const QVariant &value)
- void **resendLastData** ()

Signals

- void **dataChanged** (const QVariant &value, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp)
- void **dataChanged** (const QByteArray &value, unsigned long dataSize, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp)
- void **connectionChanged** (QCaConnectionInfo &connectionInfo)

Public Member Functions

- **QCaObject** (const QString &recordName, QObject ∗eventObject, unsigned char signalsToSendIn=SIG_VARIANT)
- **QCaObject** (const QString &recordName, QObject ∗eventObject, UserMessage ∗userMessageIn, unsigned char signalsToSendIn=SIG_VARIANT)
- bool **subscribe** ()
- bool **singleShotRead** ()
- bool **dataTypeKnown** ()
- bool **createChannel** ()
- void **deleteChannel** ()
- bool **createSubscription** ()
- bool **getChannel** ()
- bool **putChannel** ()
- bool **isChannelConnected** ()
- void **startConnectionTimer** ()
- void **stopConnectionTimer** ()
- void **setUserMessage** (UserMessage ∗userMessageIn)
- void **enableWriteCallbacks** (bool enable)
- bool **isWriteCallbacksEnabled** ()
- QString **getEgu** ()
- QStringList **getEnumerations** ()
- unsigned int **getPrecision** ()
- double **getDisplayLimitUpper** ()
- double **getDisplayLimitLower** ()
- double **getAlarmLimitUpper** ()
- double **getAlarmLimitLower** ()
- double **getWarningLimitUpper** ()
- double **getWarningLimitLower** ()
- double **getControlLimitUpper** ()
- double **getControlLimitLower** ()
- generic::generic_types **getDataType** ()

Static Public Member Functions

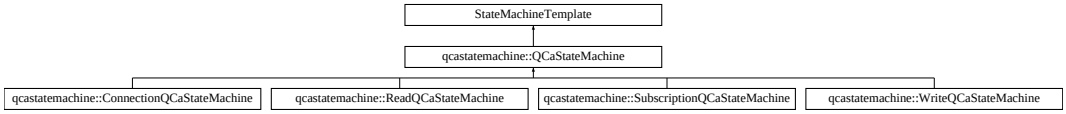
- static void **processEventStatic** (QCaEventUpdate ∗dataUpdateEvent)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QCaObject.h
- /home/andrew/epicsqt/framework/data/src/QCaObject.cpp

9.47 qcastatemachine::QCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::QCaStateMachine:



Public Member Functions

- **QCaStateMachine** (void ∗parent)
- virtual bool **process** (int requestedState)=0

Public Attributes

- QMutex **lock**
- bool **pending**
- bool **active**
- bool **expired**
- void ∗ **myWorker**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QCaStateMachine.h
- /home/andrew/epicsqt/framework/data/src/QCaStateMachine.cpp

9.48 QCaVariableNamePropertyManager Class Reference

Signals

- void **newVariableNameProperty** (QString variable, QString Substitutions, unsigned int variableIndex)

Public Member Functions

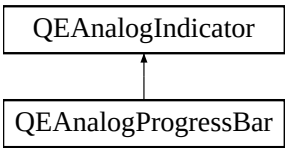
- QString **getVariableNameProperty** ()
- void **setVariableNameProperty** (QString variableNamePropertyIn)
- QString **getSubstitutionsProperty** ()
- void **setSubstitutionsProperty** (QString substitutionsPropertyIn)
- void **setVariableIndex** (unsigned int variableIndexIn)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QCaVariableNameProperty-Manager.h
- /home/andrew/epicsqt/framework/data/src/QCaVariableNamePropertyManager.-cpp

9.49 QEAnalogIndicator Class Reference

Inheritance diagram for QEAnalogIndicator:



Classes

- struct Band
- class BandList

Public Types

- enum **Orientations** { **Left_To_Right**, **Top_To_Bottom**, **Right_To_Left**, **Bottom-_To_Top** }
- enum **Modes** { **Bar**, **Scale**, **Meter** }

Public Slots

- void **setRange** (const double MinimumIn, const double MaximumIn)
- void **setValue** (const double ValueIn)

Public Member Functions

- **QEAnalogIndicator** (QWidget *parent=0)
- virtual QSize **sizeHint** () const
- double **getValue** ()
- void **setMinimum** (const double value)
- double **getMinimum** ()
- void **setMaximum** (const double value)
- double **getMaximum** ()
- void **setOrientation** (const enum Orientations value)
- enum Orientations **getOrientation** ()
- void **setMode** (const enum Modes value)
- enum Modes **getMode** ()
- void **setCentreAngle** (const int value)
- int **getCentreAngle** ()
- void **setSpanAngle** (const int value)
- int **getSpanAngle** ()
- void **setMinorInterval** (const double value)
- double **getMinorInterval** ()
- void **setMajorInterval** (const double value)
- double **getMajorInterval** ()
- void **setLogScaleInterval** (const int value)
- int **getLogScaleInterval** ()
- void **setBorderColour** (const QColor value)
- QColor **getBorderColour** ()
- void **setForegroundColour** (const QColor value)
- QColor **getForegroundColour** ()
- void **setBackgroundColour** (const QColor value)
- QColor **getBackgroundColour** ()
- void **setFontColour** (const QColor value)
- QColor **getFontColour** ()
- void **setShowText** (const bool value)
- bool **getShowText** ()
- void **setShowScale** (const bool value)
- bool **getShowScale** ()
- void **setLogScale** (const bool value)
- bool **getLogScale** ()

Protected Member Functions

- virtual QString **getTextImage** ()
- virtual BandList **getBandList** ()

Properties

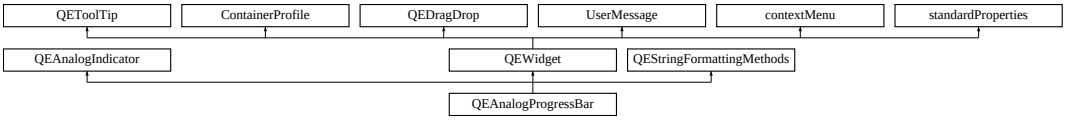
- double **value**
- double **minimum**
- double **maximum**
- double **minorInterval**
- double **majorInterval**
- int **logScaleInterval**
- bool **showText**
- bool **showScale**
- bool **logScale**
- Modes **mode**
- Orientations **orientation**
- int **centreAngle**
- int **spanAngle**
- QColor **borderColour**
- QColor **backgroundColour**
- QColor **foregroundColour**
- QColor **fontColour**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEAnalogIndicator/QEAnalogIndicator.-h
- /home/andrew/epicsqt/framework/widgets/QEAnalogIndicator/QEAnalogIndicator.-cpp

9.50 QEAnalogProgressBar Class Reference

Inheritance diagram for QEAnalogProgressBar:



Public Types

- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

- enum Formats { **Default** = QEStringFormatting::FORMAT_DEFAULT, **Floating** = QEStringFormatting::FORMAT_FLOATING, **Integer** = QEStringFormatting::FORMAT_INTEGER, **UnsignedInteger** = QEStringFormatting::FORMAT_UNSIGNEDINTEGER, **Time** = QEStringFormatting::FORMAT_TIME, **LocalEnumeration** = QEStringFormatting::FORMAT_LOCAL_ENUMERATE }

  User friendly enumerations for format property - refer to QEStringFormatting::formats for details.
- enum Notations { **Fixed** = QEStringFormatting::NOTATION_FIXED, **Scientific** = QEStringFormatting::NOTATION_SCIENTIFIC, **Automatic** = QEStringFormatting::NOTATION_AUTOMATIC }

  User friendly enumerations for notation property - refer to QEStringFormatting::notations for details.
- enum ArrayActions { **Append** = QEStringFormatting::APPEND, **Ascii** = QEStringFormatting::ASCII, **Index** = QEStringFormatting::INDEX }

  User friendly enumerations for arrayAction property - refer to QEStringFormatting::arrayActions for details.
- enum **AlarmSeverityDisplayModes** { **none**, **foreground**, **background** }

Public Slots

- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const double &out)
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.

Public Member Functions

- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.

- void setFormatProperty (Formats format)

  Access function for 'format' property - refer to 'format' property for details.

- Formats getFormatProperty ()

  Access function for 'format' property - refer to 'format' property for details.

- void setNotationProperty (Notations notation)

  Access function for 'notation' property - refer to 'notation' property for details.

- Notations getNotationProperty ()

  Access function for 'notation' property - refer to 'notation' property for details.

- void setArrayActionProperty (ArrayActions arrayAction)

  Access function for 'arrayAction' property - refer to 'arrayAction' property for details.

- ArrayActions getArrayActionProperty ()

  Access function for 'arrayAction' property - refer to 'arrayAction' property for details.

- **QEAnalogProgressBar** (QWidget ∗parent=0)
- **QEAnalogProgressBar** (const QString &variableName, QWidget ∗parent=0)
- void **setVariableNameAndSubstitutions** (QString variableNameIn, QString variableNameSubstitutionsIn, unsigned int variableIndex)
- void **setUseDbDisplayLimits** (bool useDbDisplayLimitsIn)
- bool **getUseDbDisplayLimits** ()
- void **setAlarmSeverityDisplayMode** (AlarmSeverityDisplayModes value)
- AlarmSeverityDisplayModes **getAlarmSeverityDisplayMode** ()

Protected Member Functions

- QString **getTextImage** ()
- BandList **getBandList** ()
- void **establishConnection** (unsigned int variableIndex)
- void **stringFormattingChange** ()
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **mousePressEvent** (QMouseEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()
- QString **copyVariable** ()
- QVariant **copyData** ()

Protected Attributes

- QEFloatingFormatting **floatingFormatting**

Properties

- QString variable
- QString variableSubstitutions
- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- int precision
- bool useDbPrecision
- bool leadingZero
- bool trailingZeros
- bool addUnits
- QString localEnumeration
- Formats format
- Notations notation
- ArrayActions arrayAction
- bool **useDbDisplayLimits**
- AlarmSeverityDisplayModes **alarmSeverityDisplayMode**

9.50.1 Member Function Documentation

9.50.1.1 void QEAnalogProgressBar::dbValueChanged (const double & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on E-PICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.50.1.2 void QEAnalogProgressBar::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.50.2 Property Documentation

9.50.2.1 bool QEAnalogProgressBar::addUnits [read, write]

If true (default), add engineering units supplied with the data.

9.50.2.2 bool QEAnalogProgressBar::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.50.2.3 ArrayActions QEAnalogProgressBar::arrayAction [read, write]

Text formatting option for array data. Default is ASCII. Options are:

- ASCII - treat array as a single text string. For example an array of three characters 'a' 'b' 'c' will be formatted as 'abc'.
- APPEND - treat array as an array of numbers and format a string containing them all with a space between each. For example, an array of three numbers 10, 11 and 12 will be formatted as '10 11 12'.
- INDEX - Extract a single item from the array. The item is then formatted as any other non array data would be. The item selected is determined by the arrayIndex property. For example, if arrayIndex property is 1, an array of three numbers 10, 11 and 12 will be formatted as '11'.

9.50.2.4 bool QEAnalogProgressBar::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.50.2.5 Formats QEAnalogProgressBar::format [read, write]

Format to apply to data. Default is 'Default' in which case the data type supplied with the data determines how the data is formatted. For all other options, an attempt is made to format the data as requested (whatever its native form).

9.50.2.6 unsigned QEAnalogProgressBar::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

Base used for when formatting integers. Default is 10 (duh!)

Index used to select a single item of data for formatting from an array of data. Default is 0. Only used when the arrayAction property is INDEX. Refer to the arrayAction property for more details.

9.50.2.7 bool QEAnalogProgressBar::leadingZero [read, write]

If true (default), always add a leading zero when formatting numbers.

9.50.2.8 QString QEAnalogProgressBar::localEnumeration [read, write]

An enumeration list used to data values. Used only when the formatting option is 'local enumeration'. Value is converted to an integer and used to select a string from this list.

9.50.2.9 Notations QEAnalogProgressBar::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.50.2.10 int QEAnalogProgressBar::precision [read, write]

Precision used when formatting floating point numbers. The default is 4. This is only used if useDbPrecision is false.

9.50.2.11 bool QEAnalogProgressBar::trailingZeros [read, write]

If true (default), always remove any trailing zeros when formatting numbers.

9.50.2.12 bool QEAnalogProgressBar::useDbPrecision [read, write]

If true (default), format floating point numbers using the precision supplied with the data. If false, the precision property is used.

9.50.2.13 UserLevels QEAnalogProgressBar::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through set-UserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.50.2.14 QString QEAnalogProgressBar::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the style-Manager class. Refer to the styleManager class for details about how this Style Sheet

string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.50.2.15 QString QEAnalogProgressBar::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the style-Manager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.50.2.16 QString QEAnalogProgressBar::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.50.2.17 UserLevels QEAnalogProgressBar::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.50.2.18 QString QEAnalogProgressBar::variable [read, write]

EPICS variable name (CA PV)

9.50.2.19 bool QEAnalogProgressBar::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.50.2.20 QString QEAnalogProgressBar::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME

= "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.50.2.21 bool QEAnalogProgressBar::visible [read, write]

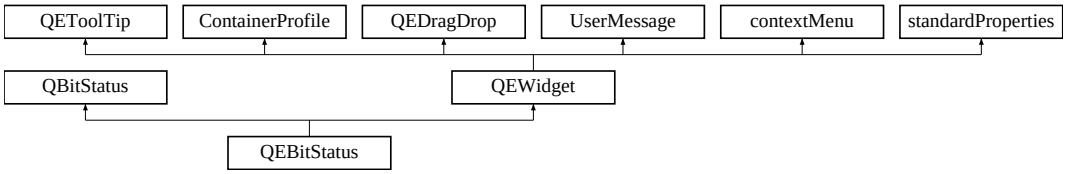
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEAnalogProgressBar/QEAnalogProgressBar.h
- /home/andrew/epicsqt/framework/widgets/QEAnalogProgressBar/QEAnalogProgressBar.cpp

9.51 QEBitStatus Class Reference

Inheritance diagram for QEBitStatus:



Public Types

- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Public Slots

- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const long &out)

Public Member Functions

- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- **QEBitStatus** (QWidget ∗parent=0)
- **QEBitStatus** (const QString &variableName, QWidget ∗parent=0)
- void **setVariableNameAndSubstitutions** (QString variableNameIn, QString variableNameSubstitutionsIn, unsigned int variableIndex)

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **mousePressEvent** (QMouseEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()
- QString **copyVariable** ()
- QVariant **copyData** ()

Protected Attributes

- QEIntegerFormatting **integerFormatting**

Properties

- QString variable
- QString variableSubstitutions
- bool variableAsToolTip
- bool enabled
- bool allowDrop

- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled

9.51.1 Member Function Documentation

9.51.1.1 void QEBitStatus::dbValueChanged (const long & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.51.1.2 void QEBitStatus::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.51.2 Property Documentation

9.51.2.1 bool QEBitStatus::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.51.2.2 bool QEBitStatus::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.51.2.3 unsigned QEBitStatus::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.51.2.4 UserLevels QEBitStatus::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.51.2.5 QString QEBitStatus::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.51.2.6 QString QEBitStatus::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.51.2.7 QString QEBitStatus::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.51.2.8 UserLevels QEBitStatus::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.51.2.9 QString QEBitStatus::variable [read, write]

EPICS variable name (CA PV)

9.51.2.10 bool QEBitStatus::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.51.2.11 QString QEBitStatus::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.51.2.12 bool QEBitStatus::visible [read, write]

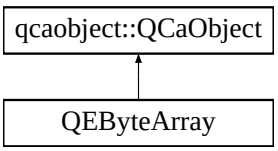
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEBitStatus/QEBitStatus.h
- /home/andrew/epicsqt/framework/widgets/QEBitStatus/QEBitStatus.cpp

9.52 QEByteArray Class Reference

Inheritance diagram for QEByteArray:



Public Slots

- void **writeByteArray** (const QByteArray &data)

Signals

- void **byteArrayConnectionChanged** (QCaConnectionInfo &connectionInfo, const unsigned int &variableIndex)
- void **byteArrayChanged** (const QByteArray &value, unsigned long dataSize, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)

Public Member Functions

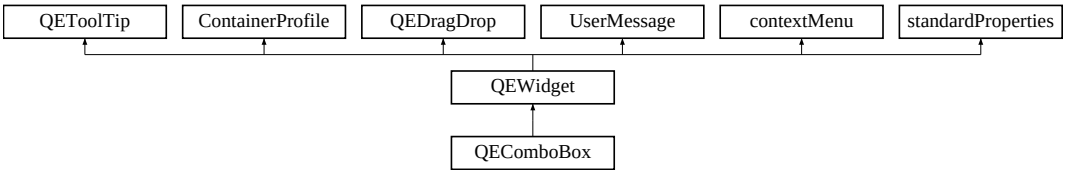
- **QEByteArray** (QString recordName, QObject *eventObject, unsigned int variableIndexIn)
- **QEByteArray** (QString recordName, QObject *eventObject, unsigned int variableIndexIn, UserMessage *userMessageIn)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QEByteArray.h
- /home/andrew/epicsqt/framework/data/src/QEByteArray.cpp

9.53 QEComboBox Class Reference

Inheritance diagram for QEComboBox:



Public Types

- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Public Slots

- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const qlonglong &out)
- void userChange (const QString &oldValue, const QString &newValue, const QString &lastValue)

  Internal use only. Used by QEConfiguredLayout to be notified when one of its widgets has written something.

Public Member Functions

- **QEComboBox** (QWidget *parent=0)
- **QEComboBox** (const QString &variableName, QWidget *parent=0)
- void **setWriteOnChange** (bool writeOnChangeIn)
- bool **getWriteOnChange** ()
- void **setSubscribe** (bool subscribe)
- bool **getSubscribe** ()
- void **setUseDbEnumerations** (bool useDbEnumerations)
- bool **getUseDbEnumerations** ()
- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent *event)
- void **dropEvent** (QDropEvent *event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Protected Attributes

- QEIntegerFormatting **integerFormatting**
- bool **useDbEnumerations**
- bool writeOnChange

Properties

- QString variable
- QString variableSubstitutions
- bool subscribe
- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled

9.53.1 Member Function Documentation

9.53.1.1 void QEComboBox::dbValueChanged (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on E-PICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.53.1.2 void QEComboBox::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.53.2 Member Data Documentation

9.53.2.1 bool QEComboBox::writeOnChange [read, write, protected]

Sets if this widget writes any changes as the user selects values (the QComboBox 'activated' signal is emitted). Default is 'true' (writes any changes when the QComboBox 'activated' signal is emitted).

9.53.3 Property Documentation

9.53.3.1 bool QEComboBox::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.53.3.2 bool QEComboBox::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.53.3.3 unsigned QEComboBox::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.53.3.4 bool QEComboBox::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.53.3.5 UserLevels QEComboBox::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.53.3.6 QString QEComboBox::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet

string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.53.3.7 QString QEComboBox::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the style-Manager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.53.3.8 QString QEComboBox::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.53.3.9 UserLevels QEComboBox::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.53.3.10 QString QEComboBox::variable [read, write]

EPICS variable name (CA PV)

9.53.3.11 bool QEComboBox::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.53.3.12 QString QEComboBox::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME

= "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.53.3.13 bool QEComboBox::visible [read, write]

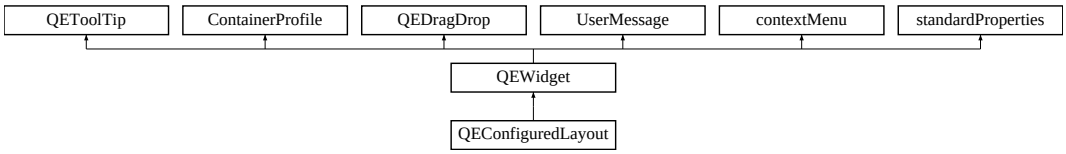
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEComboBox/QEComboBox.h
- /home/andrew/epicsqt/framework/widgets/QEComboBox/QEComboBox.cpp

9.54 QEConfiguredLayout Class Reference

Inheritance diagram for QEConfiguredLayout:



Public Types

- enum **configurationTypesProperty** { **File** = FROM_FILE, **Text** = FROM_TEXT }
- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **userTypesProperty** { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

Public Member Functions

- **QEConfiguredLayout** (QWidget ∗pParent=0, bool pSubscription=true)
- void **setItemDescription** (QString pValue)
- QString **getItemDescription** ()
- void **setShowItemList** (bool pValue)
- bool **getShowItemList** ()
- void **setConfigurationType** (int pValue)
- int **getConfigurationType** ()
- void **setConfigurationFile** (QString pValue)
- QString **getConfigurationFile** ()

- void **setConfigurationText** (QString pValue)
- QString **getConfigurationText** ()
- void **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- void **setCurrentUserType** (int pValue)
- int **getCurrentUserType** ()
- void **refreshFields** ()
- void **userLevelChanged** (userLevels pValue)
- void **setConfigurationTypeProperty** (configurationTypesProperty pConfiguration-Type)
- configurationTypesProperty **getConfigurationTypeProperty** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()
- void **setCurrentUserTypeProperty** (userTypesProperty pUserType)
- userTypesProperty **getCurrentUserTypeProperty** ()

Public Attributes

- QList< _Item * > **itemList**
- QList< _Field * > **currentFieldList**

Protected Attributes

- QLabel * **qLabelItemDescription**
- QComboBox * **qComboBoxItemList**
- QVBoxLayout * **qVBoxLayoutFields**
- QScrollArea * **qScrollArea**
- QString **configurationFile**
- QString **configurationText**
- int **configurationType**
- int **detailsLayout**
- int **currentUserType**
- bool **subscription**

Properties

- QString **itemDescription**
- bool **showItemList**
- configurationTypesProperty **configurationType**
- detailsLayoutProperty **detailsLayout**
- userTypesProperty **currentUserType**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEConfiguredLayout/QEConfigured-Layout.h
- /home/andrew/epicsqt/framework/widgets/QEConfiguredLayout/QEConfigured-Layout.cpp

9.55 QEConfiguredLayoutManager Class Reference

Public Member Functions

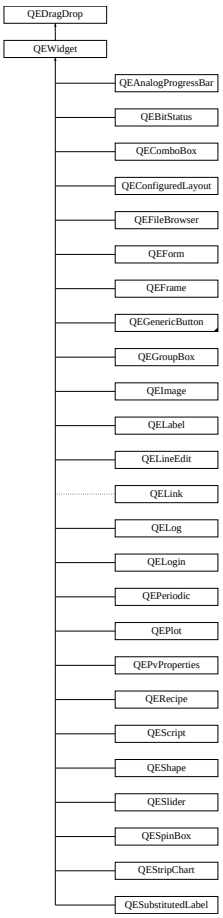
- **QEConfiguredLayoutManager** (QObject ∗pParent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget ∗ **createWidget** (QWidget ∗pParent)
- void **initialize** (QDesignerFormEditorInterface ∗pCore)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEConfiguredLayout/QEConfigured-LayoutManager.h
- /home/andrew/epicsqt/framework/widgets/QEConfiguredLayout/QEConfigured-LayoutManager.cpp

9.56 QEDragDrop Class Reference

Inheritance diagram for QEDragDrop:



Public Member Functions

- **QEDragDrop** (QWidget ∗ownerIn)

Protected Member Functions

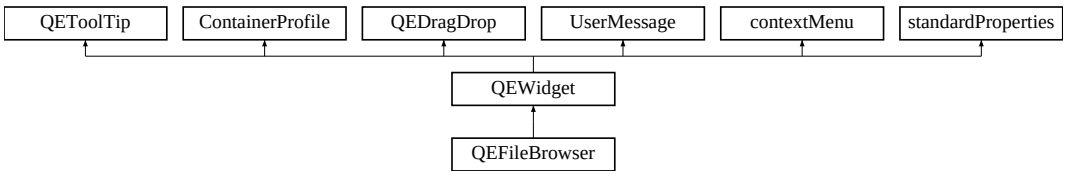
- void **qcaDragEnterEvent** (QDragEnterEvent ∗event)
- void **qcaDropEvent** (QDropEvent ∗event)
- void **qcaMousePressEvent** (QMouseEvent ∗event)
- virtual void **setDrop** (QVariant)
- virtual QVariant **getDrop** ()
- void **setAllowDrop** (bool allowDropIn)
- bool **getAllowDrop** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/QEDragDrop.h
- /home/andrew/epicsqt/framework/widgets/src/QEDragDrop.cpp

9.57 QEFileBrowser Class Reference

Inheritance diagram for QEFileBrowser:



Public Types

- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }

Signals

- void **selected** (QString pFilename)

Public Member Functions

- **QEFileBrowser** (QWidget ∗pParent=0)
- void **setDirectoryPath** (QString pValue)
- QString **getDirectoryPath** ()
- void **setShowDirectoryPath** (bool pValue)
- bool **getShowDirectoryPath** ()
- void **setShowDirectoryBrowser** (bool pValue)
- bool **getShowDirectoryBrowser** ()
- void **setShowRefresh** (bool pValue)
- bool **getShowRefresh** ()
- void **setShowColumnTime** (bool pValue)
- bool **getShowColumnTime** ()
- void **setShowColumnSize** (bool pValue)
- bool **getShowColumnSize** ()
- void **setShowColumnFilename** (bool pValue)
- bool **getShowColumnFilename** ()
- void **setShowFileExtension** (bool pValue)
- bool **getShowFileExtension** ()
- void **setFileFilter** (QString pValue)
- QString **getFileFilter** ()
- void **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- void **updateTable** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()

Protected Attributes

- QLineEdit * **qlineEditDirectoryPath**
- QPushButton * **qPushButtonDirectoryBrowser**
- QPushButton * **qPushButtonRefresh**
- _QTableWidgetFileBrowser * **qTableWidgetFileBrowser**
- QString **fileFilter**
- bool **showFileExtension**
- int **detailsLayout**

Properties

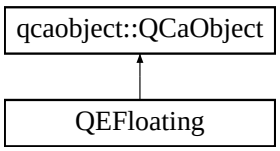
- QString **directoryPath**
- bool **showDirectoryPath**
- bool **showDirectoryBrowser**
- bool **showRefresh**
- bool **showColumnTime**
- bool **showColumnSize**
- bool **showColumnFilename**
- detailsLayoutProperty **detailsLayout**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEFileBrowser/QEFileBrowser.h
- /home/andrew/epicsqt/framework/widgets/QEFileBrowser/QEFileBrowser.cpp

9.58 QEFloating Class Reference

Inheritance diagram for QEFloating:



Public Slots

- void **writeFloating** (const double &data)

Signals

- void **floatingConnectionChanged** (QCaConnectionInfo &connectionInfo, const unsigned int &variableIndex)
- void **floatingChanged** (const double &value, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)
- void **floatingArrayChanged** (const QVector< double > &values, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)

Public Member Functions

- **QEFloating** (QString recordName, QObject *eventObject, QEFloatingFormatting *floatingFormattingIn, unsigned int variableIndexIn)
- **QEFloating** (QString recordName, QObject *eventObject, QEFloatingFormatting *floatingFormattingIn, unsigned int variableIndexIn, UserMessage *userMessageIn)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QEFloating.h
- /home/andrew/epicsqt/framework/data/src/QEFloating.cpp

9.59 QEFloatingFormatting Class Reference

Public Types

- enum **formats** { **FORMAT_e** = 'e', **FORMAT_E** = 'E', **FORMAT_f** = 'f', **FORMAT_g** = 'g', **FORMAT_G** = 'G' }

Public Member Functions

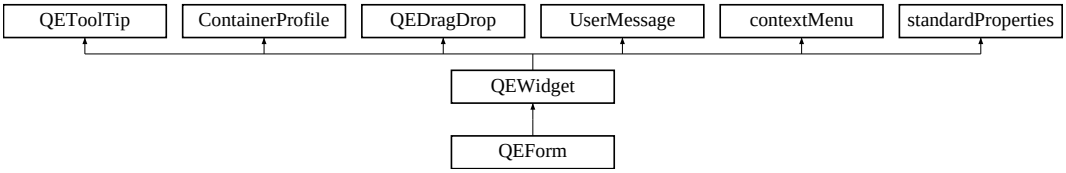
- double **formatFloating** (const QVariant &value)
- QVector< double > **formatFloatingArray** (const QVariant &value)
- QVariant **formatValue** (const double &floatingValue, generic::generic_types valueType)
- void **setPrecision** (unsigned int precision)
- void **setFormat** (formats format)
- unsigned int **getPrecision** ()
- int **getFormat** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QEFloatingFormatting.h
- /home/andrew/epicsqt/framework/data/src/QEFloatingFormatting.cpp

9.60 QEForm Class Reference

Inheritance diagram for QEForm:



Public Types

- enum **creationOptions** { **CREATION_OPTION_OPEN**, **CREATION_OPTION_-NEW_TAB**, **CREATION_OPTION_NEW_WINDOW** }
- enum **MessageFilterOptions** { **Match** = UserMessage::MESSAGE_FILTER_M-ATCH, **None** = UserMessage::MESSAGE_FILTER_NONE }

Public Slots

- bool **readUiFile** ()
- void **launchGui** (QString guiName, QEForm::creationOptions createOption)

Public Member Functions

- **QEForm** (QWidget ∗parent=0)
- **QEForm** (const QString &uifileNameIn, QWidget ∗parent=0)
- void **commonInit** (const bool alertIfUINoFoundIn)
- QString **getQEGuiTitle** ()
- QString **getFullFileName** ()
- void **setVariableNameAndSubstitutions** (QString variableNameIn, QString variableNameSubstitutionsIn, unsigned int variableIndex)
- void **setUiFileName** (QString uiFile)
- QString **getUiFileName** ()
- void **setHandleGuiLaunchRequests** (bool handleGuiLaunchRequests)
- bool **getHandleGuiLaunchRequests** ()
- void **setResizeContents** (bool resizeContentsIn)
- bool **getResizeContents** ()
- void **setVariableNameSubstitutionsProperty** (QString variableName-Substitutions)
- QString **getVariableNameSubstitutionsProperty** ()
- MessageFilterOptions **getMessageFormFilter** ()
- void **setMessageFormFilter** (MessageFilterOptions messageFormFilter)
- MessageFilterOptions **getMessageSourceFilter** ()
- void **setMessageSourceFilter** (MessageFilterOptions messageSourceFilter)

Protected Member Functions

- void **setVariableNameSubstitutions** (QString variableNameSubstitutionsIn)

Protected Attributes

- QString **uiFileName**
- QString **fullUiFileName**
- bool **handleGuiLaunchRequests**
- bool **resizeContents**

Properties

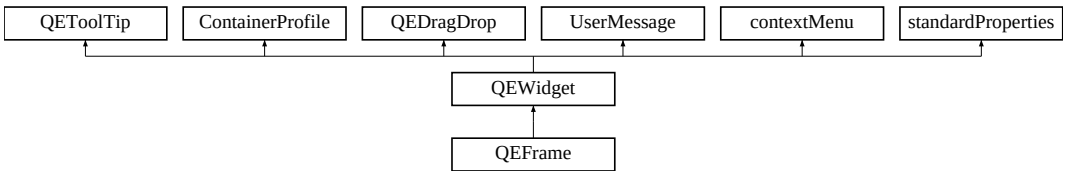
- QString **uiFile**
- QString **variableSubstitutions**
- unsigned **int**
- MessageFilterOptions **messageFormFilter**
- MessageFilterOptions **messageSourceFilter**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEForm/QEForm.h
- /home/andrew/epicsqt/framework/widgets/QEForm/QEForm.cpp

9.61 QEFrame Class Reference

Inheritance diagram for QEFrame:



Public Types

- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Public Slots

- void requestEnabled (const bool &state)

Public Member Functions

- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- **QEFrame** (QWidget *parent=0)
- QSize **sizeHint** () const

Properties

- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled

9.61.1 Member Function Documentation

9.61.1.1 void QEFrame::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.61.2 Property Documentation

9.61.2.1 bool QEFrame::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.61.2.2 bool QEFrame::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.61.2.3 unsigned QEFrame::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.61.2.4 UserLevels QEFrame::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.61.2.5 QString QEFrame::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.61.2.6 QString QEFrame::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the style-Manager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.61.2.7 QString QEFrame::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.61.2.8 UserLevels QEFrame::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.61.2.9 bool QEFrame::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.61.2.10 bool QEFrame::visible [read, write]

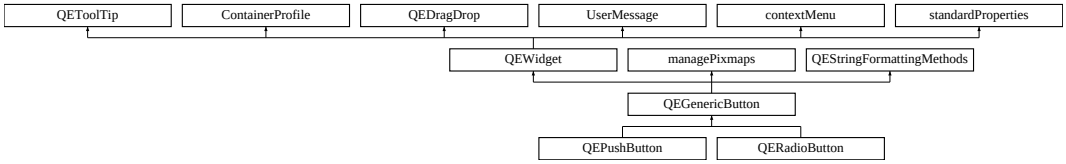
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEFrame/QEFrame.h
- /home/andrew/epicsqt/framework/widgets/QEFrame/QEFrame.cpp

9.62 QEGenericButton Class Reference

Inheritance diagram for QEGenericButton:



Public Types

- enum **updateOptions** { **UPDATE_TEXT**, **UPDATE_ICON**, **UPDATE_TEXT_AND_ICON**, **UPDATE_STATE** }

Public Member Functions

- **QEGenericButton** (QWidget ∗owner)
- void **setSubscribe** (bool subscribe)
- bool **getSubscribe** ()
- void **setUpdateOption** (updateOptions updateOptionIn)
- updateOptions **getUpdateOption** ()
- void **setTextAlignment** (Qt::Alignment alignment)
- Qt::Alignment **getTextAlignment** ()
- void **setPassword** (QString password)
- QString **getPassword** ()
- void **setConfirmAction** (bool confirmRequiredIn)
- bool **getConfirmAction** ()
- void **setWriteOnPress** (bool writeOnPress)
- bool **getWriteOnPress** ()
- void **setWriteOnRelease** (bool writeOnRelease)
- bool **getWriteOnRelease** ()
- void **setWriteOnClick** (bool writeOnClick)
- bool **getWriteOnClick** ()
- void **setPressText** (QString pressText)
- QString **getPressText** ()
- void **setReleaseText** (QString releaseTextIn)
- QString **getReleaseText** ()
- void **setClickText** (QString clickTextIn)
- QString **getClickText** ()
- void **setClickCheckedText** (QString clickCheckedTextIn)
- QString **getClickCheckedText** ()
- void **setProgram** (QString program)
- QString **getProgram** ()
- void **setArguments** (QStringList arguments)

- QStringList **getArguments** ()
- void **setGuiName** (QString guiName)
- QString **getGuiName** ()
- void **setCreationOption** (QEForm::creationOptions creationOption)
- QEForm::creationOptions **getCreationOption** ()
- void **setLabelTextProperty** (QString labelTextIn)
- QString **getLabelTextProperty** ()
- void **onGeneralMessage** (QString message)

Protected Member Functions

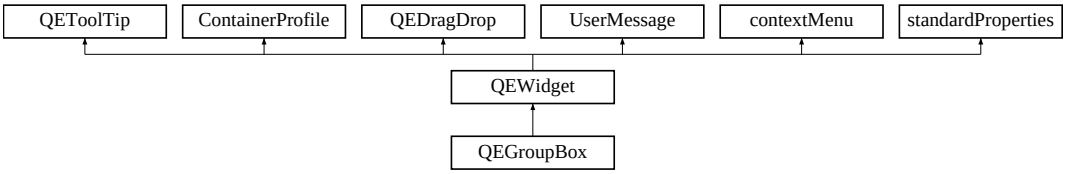
- void **connectionChanged** (QCaConnectionInfo &connectionInfo)
- void **setGenericButtonText** (const QString &text, QCaAlarmInfo &alarmInfo, QCaDateTime &, const unsigned int &variableIndex)
- void **userPressed** ()
- void **userReleased** ()
- void **userClicked** (bool checked)
- void **launchGui** (QString guiName, QEForm::creationOptions creationOption)
- virtual updateOptions **getDefaultUpdateOption** ()=0
- void **setup** ()
- void **establishConnection** (unsigned int variableIndex)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEButton/QEGenericButton.h
- /home/andrew/epicsqt/framework/widgets/QEButton/QEGenericButton.cpp

9.63 QEGroupBox Class Reference

Inheritance diagram for QEGroupBox:



Public Types

- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Public Slots

- void requestEnabled (const bool &state)

Public Member Functions

- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- **QEGroupBox** (QWidget *parent=0)
- QSize **sizeHint** () const

Properties

- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled

9.63.1 Member Function Documentation

9.63.1.1 void QEGroupBox::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.63.2 Property Documentation

9.63.2.1 bool QEGroupBox::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.63.2.2 bool QEGroupBox::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.63.2.3 unsigned QEGroupBox::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.63.2.4 UserLevels QEGroupBox::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.63.2.5 QString QEGroupBox::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.63.2.6 QString QEGroupBox::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.63.2.7 QString QEGroupBox::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.63.2.8 UserLevels QEGroupBox::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.63.2.9 bool QEGroupBox::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.63.2.10 bool QEGroupBox::visible [read, write]

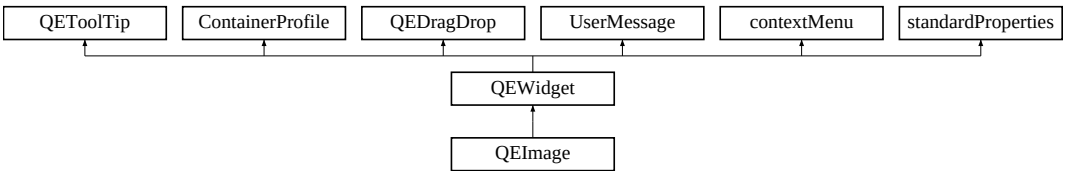
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEGroupBox/QEGroupBox.h
- /home/andrew/epicsqt/framework/widgets/QEGroupBox/QEGroupBox.cpp

9.64 QEImage Class Reference

Inheritance diagram for QEImage:



Public Types

- enum **selectOptions** { **SO_NONE**, **SO_PANNING**, **SO_VSLICE**, **SO_HSLICE**, **SO_AREA**, **SO_PROFILE**, **SO_TARGET**, **SO_BEAM** }
- enum **formatOptions** { **GREY8**, **GREY12**, **GREY16**, **RGB_888** }
- enum **resizeOptions** { **RESIZE_OPTION_ZOOM**, **RESIZE_OPTION_FIT** }
- enum **rotationOptions** { **ROTATION_0**, **ROTATION_90_RIGHT**, **ROTATION_-90_LEFT**, **ROTATION_180** }
- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.
- enum **FormatOptions** { **Grey_8** = QEImage::GREY8, **Grey_12** = QEImage::GREY12, **Grey_16** = QEImage::GREY16, **RGB** = QEImage::RGB_888 }
- enum **ResizeOptions** { **Zoom** = QEImage::RESIZE_OPTION_ZOOM, **Fit** = QEImage::RESIZE_OPTION_FIT }
- enum **RotationOptions** { **NoRotation** = QEImage::ROTATION_0, **Rotate90-Right** = QEImage::ROTATION_90_RIGHT, **Rotate90Left** = QEImage::ROTATION_90_LEFT, **Rotate180** = QEImage::ROTATION_180 }

Public Slots

- void **setSelectPanMode** ()
- void **setSelectVSliceMode** ()
- void **setSelectHSliceMode** ()
- void **setSelectAreaMode** ()
- void **setSelectProfileMode** ()
- void **setSelectTargetMode** ()
- void **setSelectBeamMode** ()
- void **pauseClicked** ()
- void **saveClicked** ()
- void **roiClicked** ()
- void **resetRoiClicked** ()
- void **targetClicked** ()
- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const QString &out)
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.

Public Member Functions

- **QEImage** (QWidget *parent=0)
- **QEImage** (const QString &variableName, QWidget *parent=0)
- selectOptions **getSelectionOption** ()
- void **setFormatOption** (formatOptions formatOption)
- formatOptions **getFormatOption** ()
- void **setResizeOption** (resizeOptions resizeOptionIn)
- resizeOptions **getResizeOption** ()
- void **setZoom** (int zoomIn)
- int **getZoom** ()
- void **setRotation** (rotationOptions rotationIn)
- rotationOptions **getRotation** ()
- void **setHorizontalFlip** (bool flipHozIn)
- bool **getHorizontalFlip** ()
- void **setVerticalFlip** (bool flipVertIn)
- bool **getVerticalFlip** ()
- void **setInitialHozScrollPos** (int initialHosScrollPosIn)
- int **getInitialHozScrollPos** ()
- void **setInitialVertScrollPos** (int initialVertScrollPosIn)
- int **getInitialVertScrollPos** ()
- void **setDisplayAcquirePeriod** (bool displayAcquirePeriodIn)
- bool **getDisplayAcquirePeriod** ()
- void **setDisplayExposureTime** (bool displayExposureTimeIn)
- bool **getDisplayExposureTime** ()
- void **setDisplayButtonBar** (bool displayButtonBarIn)
- bool **getDisplayButtonBar** ()
- void **setShowTime** (bool pValue)
- bool **getShowTime** ()
- void **setVertSliceMarkupColor** (QColor pValue)
- QColor **getVertSliceMarkupColor** ()
- void **setHozSliceMarkupColor** (QColor pValue)
- QColor **getHozSliceMarkupColor** ()
- void **setProfileMarkupColor** (QColor pValue)
- QColor **getProfileMarkupColor** ()
- void **setAreaMarkupColor** (QColor pValue)
- QColor **getAreaMarkupColor** ()
- void **setTargetMarkupColor** (QColor pValue)
- QColor **getTargetMarkupColor** ()

- void **setBeamMarkupColor** (QColor pValue)
- QColor **getBeamMarkupColor** ()
- void **setTimeMarkupColor** (QColor pValue)
- QColor **getTimeMarkupColor** ()
- void **setDisplayCursorPixelInfo** (bool displayCursorPixelInfoIn)
- bool **getDisplayCursorPixelInfo** ()
- void **setContrastReversal** (bool contrastReversalIn)
- bool **getContrastReversal** ()
- void **setEnablePan** (bool enablePanIn)
- bool **getEnablePan** ()
- void **setEnableVertSliceSelection** (bool enableVSliceSelectionIn)
- bool **getEnableVertSliceSelection** ()
- void **setEnableHozSliceSelection** (bool enableHSliceSelectionIn)
- bool **getEnableHozSliceSelection** ()
- void **setEnableAreaSelection** (bool enableAreaSelectionIn)
- bool **getEnableAreaSelection** ()
- void **setEnableProfileSelection** (bool enableProfileSelectionIn)
- bool **getEnableProfileSelection** ()
- void **setEnableTargetSelection** (bool enableTargetSelectionIn)
- bool **getEnableTargetSelection** ()
- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void **setFormatOptionProperty** (FormatOptions formatOption)
- FormatOptions **getFormatOptionProperty** ()
- void **setResizeOptionProperty** (ResizeOptions resizeOption)
- ResizeOptions **getResizeOptionProperty** ()
- void **setRotationProperty** (RotationOptions rotation)
- RotationOptions **getRotationProperty** ()

Protected Types

- enum **variableIndexes** { **IMAGE_VARIABLE**, **WIDTH_VARIABLE**, **HEIGHT_VARIABLE**, **ROI_X_VARIABLE**, **ROI_Y_VARIABLE**, **ROI_W_VARIABLE**, **ROI_H_VARIABLE**, **TARGET_X_VARIABLE**, **TARGET_Y_VARIABLE**, **BEAM_X_VARIABLE**, **BEAM_Y_VARIABLE**, **TARGET_TRIGGER_VARIABLE**, **CLIPPING_ONOFF_VARIABLE**, **CLIPPING_LOW_VARIABLE**, **CLIPPING_HIGH_VARIABLE**, **QEIMAGE_NUM_VARIABLES** }

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant v)
- void **resizeEvent** (QResizeEvent ∗)

Protected Attributes

- QEIntegerFormatting **integerFormatting**
- resizeOptions **resizeOption**
- int **zoom**
- rotationOptions **rotation**
- bool **flipVert**
- bool **flipHoz**
- int **initialHozScrollPos**
- int **initialVertScrollPos**
- bool **displayButtonBar**

Properties

- QString imageVariable
- QString widthVariable
- QString heightVariable
- QString regionOfInterestXVariable
- QString regionOfInterestYVariable
- QString regionOfInterestWVariable
- QString regionOfInterestHVariable
- QString targetXVariable
- QString targetYVariable
- QString beamXVariable

- QString beamYVariable
- QString targetTriggerVariable
- QString clippingOnOffVariable
- QString clippingLowVariable
- QString clippingHighVariable
- QString variableSubstitutions
- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- FormatOptions **formatOption**
- bool **enableVertSliceSelection**
- bool **enableHozSliceSelection**
- bool **showTime**
- QColor **vertSliceColor**
- QColor **hozSliceColor**
- QColor **profileColor**
- QColor **areaColor**
- QColor **beamColor**
- QColor **targetColor**
- QColor **timeColor**
- ResizeOptions **resizeOption**
- RotationOptions **rotation**
- bool **verticalFlip**
- bool **horizontalFlip**
- int **initialHosScrollPos**

9.64.1 Member Function Documentation

9.64.1.1 void QEImage::dbValueChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.64.1.2 void QEImage::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.64.2 Property Documentation

9.64.2.1 bool QEImage::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.64.2.2 QString QEImage::beamXVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the selected beam X position.

9.64.2.3 QString QEImage::beamYVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the selected beam Y position.

9.64.2.4 QString QEImage::clippingHighVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector clipping high level.

9.64.2.5 QString QEImage::clippingLowVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector clipping low level.

9.64.2.6 QString QEImage::clippingOnOffVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector clipping on/off command.

9.64.2.7 bool QEImage::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.64.2.8 QString QEImage::heightVariable [read, write]

EPICS variable name (CA PV). This variable is used to read the height of the image.

9.64.2.9 QString QEImage::imageVariable [read, write]

EPICS variable name (CA PV). This variable is used as the source the image waveform.

9.64.2.10 unsigned QEImage::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.64.2.11 QString QEImage::regionOfInterestHVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the region of interest height.

9.64.2.12 QString QEImage::regionOfInterestWVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the region of interest width.

9.64.2.13 QString QEImage::regionOfInterestXVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the region of interest X position.

9.64.2.14 QString QEImage::regionOfInterestYVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the region of interest Y position.

9.64.2.15 QString QEImage::targetTriggerVariable [read, write]

EPICS variable name (CA PV). This variable is used to write a 'trigger' to initiate movement of the target into the beam as defined by the target and beam X and Y positions.

9.64.2.16 QString QEImage::targetXVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the selected target X position.

9.64.2.17 QString QEImage::targetYVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the selected target Y position.

9.64.2.18 UserLevels QEImage::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through set-UserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.64.2.19 QString QEImage::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the style-Manager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.64.2.20 QString QEImage::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the style-Manager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.64.2.21 QString QEImage::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.64.2.22 UserLevels QEImage::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is

set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.64.2.23 bool QEImage::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.64.2.24 QString QEImage::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'CAM=1, NAME = "Image 1"' These substitutions are applied to all the variable names.

9.64.2.25 bool QEImage::visible [read, write]

Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

9.64.2.26 QString QEImage::widthVariable [read, write]

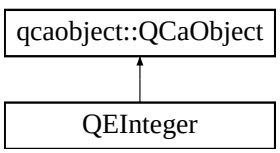
EPICS variable name (CA PV). This variable is used to read the width of the image.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/QEImage.h
- /home/andrew/epicsqt/framework/widgets/QEImage/QEImage.cpp

9.65 QEInteger Class Reference

Inheritance diagram for QEInteger:



Public Slots

- void **writeInteger** (const long &data)

Signals

- void **integerConnectionChanged** (QCaConnectionInfo &connectionInfo, const unsigned int &variableIndex)
- void **integerChanged** (const long &value, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)
- void **integerArrayChanged** (const QVector< long > &values, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)

Public Member Functions

- **QEInteger** (QString recordName, QObject *eventObject, QEIntegerFormatting *integerFormattingIn, unsigned int variableIndexIn)
- **QEInteger** (QString recordName, QObject *eventObject, QEIntegerFormatting *integerFormattingIn, unsigned int variableIndexIn, UserMessage *userMessageIn)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QEInteger.h
- /home/andrew/epicsqt/framework/data/src/QEInteger.cpp

9.66 QEIntegerFormatting Class Reference

`#include <QEIntegerFormatting.h>`

Public Member Functions

- QEIntegerFormatting ()

  Constructor.
- long formatInteger (const QVariant &value)
- QVector< long > formatIntegerArray (const QVariant &value)
- QVariant formatValue (const long &integerValue, generic::generic_types valueType)
- void setRadix (unsigned int radix)

  Set the radix used for all conversions. Default is 10.
- unsigned int getPrecision ()

  Get the precision used for all conversions.
- unsigned int getRadix ()

  Get the radix used for all conversions.

9.66.1 Detailed Description

This class holds formatting instructions and uses them to convert between an integer and a QVariant of any type. It is generally set up with it's formatting instructions and then passed to a QEInteger class that will sink and source integer data to widgets or other code. It is used to convert data to and from a QCaObject (which sources and sinks data in the form of a QVariant where the QVariant reflects the underlying variable data type) and the QEInteger class. An example of a requirement for integer data is a combo box which must determine an integer index to select a menu option.

9.66.2 Member Function Documentation

9.66.2.1 long QEIntegerFormatting::formatInteger (const QVariant & *value*)

Given a data value of any type, format it as an integer according to the formatting instructions held by the class. This is used to convert the QVariant value received from a QCaObject, which is still based on the data variable type, to an integer.

9.66.2.2 QVector< long > QEIntegerFormatting::formatIntegerArray (const QVariant & *value*)

Given a data value of any type, format it as an array of integers according to the formatting instructions held by the class. This is used to convert the QVariant value received from a QCaObject, which is still based on the data variable type, to an integer array. Typically used where the input QVariant value is an array of data values, but will work for any QVariant type.

9.66.2.3 QVariant QEIntegerFormatting::formatValue (const long & *integerValue,* generic::generic_types *valueType*)

Given an integer value, format it as a data value of the specified type, according to the formatting instructions held by the class. This is used when writing integer data to a QCaObject.

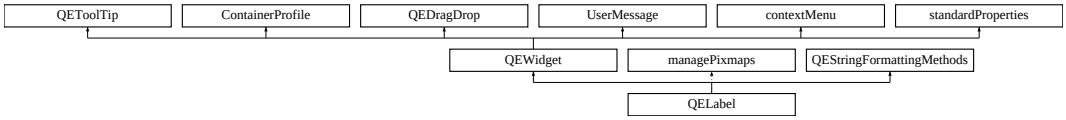
The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QEIntegerFormatting.h
- /home/andrew/epicsqt/framework/data/src/QEIntegerFormatting.cpp

9.67 QELabel Class Reference

`#include <QELabel.h>`

Inheritance diagram for QELabel:



Public Types

- enum updateOptions { UPDATE_TEXT, UPDATE_PIXMAP }
- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.
- enum Formats { **Default** = QEStringFormatting::FORMAT_DEFAULT, **Floating** = QEStringFormatting::FORMAT_FLOATING, **Integer** = QEStringFormatting::FORMAT_INTEGER, **UnsignedInteger** = QEStringFormatting::FORMAT_UNSIGNEDINTEGER, **Time** = QEStringFormatting::FORMAT_TIME, **LocalEnumeration** = QEStringFormatting::FORMAT_LOCAL_ENUMERATE }

  User friendly enumerations for format property - refer to QEStringFormatting::formats for details.
- enum Notations { **Fixed** = QEStringFormatting::NOTATION_FIXED, **Scientific** = QEStringFormatting::NOTATION_SCIENTIFIC, **Automatic** = QEStringFormatting::NOTATION_AUTOMATIC }

  User friendly enumerations for notation property - refer to QEStringFormatting::notations for details.
- enum ArrayActions { **Append** = QEStringFormatting::APPEND, **Ascii** = QEStringFormatting::ASCII, **Index** = QEStringFormatting::INDEX }

  User friendly enumerations for arrayAction property - refer to QEStringFormatting::arrayActions for details.
- enum UpdateOptions { **Text** = QELabel::UPDATE_TEXT, **Picture** = QELabel::UPDATE_PIXMAP }

  User friendly enumerations for updateOption property - refer to QELabel::updateOptions for details.

Public Slots

- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const QString &out)
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.

Public Member Functions

- QELabel (QWidget *parent=0)
- QELabel (const QString &variableName, QWidget *parent=0)
- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setFormatProperty (Formats format)

  Access function for 'format' property - refer to 'format' property for details.
- Formats getFormatProperty ()

  Access function for 'format' property - refer to 'format' property for details.
- void setNotationProperty (Notations notation)

  Access function for 'notation' property - refer to 'notation' property for details.
- Notations getNotationProperty ()

  Access function for 'notation' property - refer to 'notation' property for details.
- void setArrayActionProperty (ArrayActions arrayAction)

  Access function for 'arrayAction' property - refer to 'arrayAction' property for details.
- ArrayActions getArrayActionProperty ()

  Access function for 'arrayAction' property - refer to 'arrayAction' property for details.
- void setUpdateOptionProperty (UpdateOptions updateOption)

  Access function for 'updateOption' property - refer to 'updateOption' property for details.
- UpdateOptions getUpdateOptionProperty ()

  Access function for 'updateOption' property - refer to 'updateOption' property for details.
- void setPixmap0Property (QPixmap pixmap)

  Access function for 'pixmap0' property - refer to 'pixmap0' property for details.
- void setPixmap1Property (QPixmap pixmap)

  Access function for 'pixmap1' property - refer to 'pixmap1' property for details.
- void setPixmap2Property (QPixmap pixmap)

  Access function for 'pixmap2' property - refer to 'pixmap2' property for details.
- void setPixmap3Property (QPixmap pixmap)

Access function for 'pixmap3' property - refer to 'pixmap3' property for details.

- void setPixmap4Property (QPixmap pixmap)

  Access function for 'pixmap4' property - refer to 'pixmap4' property for details.
- void setPixmap5Property (QPixmap pixmap)

  Access function for 'pixmap5' property - refer to 'pixmap5' property for details.
- void setPixmap6Property (QPixmap pixmap)

  Access function for 'pixmap6' property - refer to 'pixmap6' property for details.
- void setPixmap7Property (QPixmap pixmap)

  Access function for 'pixmap7' property - refer to 'pixmap7' property for details.
- QPixmap getPixmap0Property ()

  Access function for 'pixmap0' property - refer to 'pixmap0' property for details.
- QPixmap getPixmap1Property ()

  Access function for 'pixmap1' property - refer to 'pixmap1' property for details.
- QPixmap getPixmap2Property ()

  Access function for 'pixmap2' property - refer to 'pixmap2' property for details.
- QPixmap getPixmap3Property ()

  Access function for 'pixmap3' property - refer to 'pixmap3' property for details.
- QPixmap getPixmap4Property ()

  Access function for 'pixmap4' property - refer to 'pixmap4' property for details.
- QPixmap getPixmap5Property ()

  Access function for 'pixmap5' property - refer to 'pixmap5' property for details.
- QPixmap getPixmap6Property ()

  Access function for 'pixmap6' property - refer to 'pixmap6' property for details.
- QPixmap getPixmap7Property ()

  Access function for 'pixmap7' property - refer to 'pixmap7' property for details.

Properties

- QString variable
- QString variableSubstitutions
- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- int precision
- bool useDbPrecision
- bool leadingZero
- bool trailingZeros

- bool addUnits
- QString localEnumeration
- Formats format
- Notations notation
- ArrayActions arrayAction
- UpdateOptions updateOption
- QPixmap pixmap0
- QPixmap pixmap1
- QPixmap pixmap2
- QPixmap pixmap3
- QPixmap pixmap4
- QPixmap pixmap5
- QPixmap pixmap6
- QPixmap pixmap7

9.67.1 Detailed Description

This class is a EPICS aware label widget based on the Qt label widget. When a variable is defined, the label text (or optionally the background pixmap) will be updated. The label will be disabled if the variable is invalid. It is tighly integrated with the base class QEWidget which provides generic support such as macro substitutions, drag/drop, and standard properties.

9.67.2 Member Enumeration Documentation

9.67.2.1 enum QELabel::updateOptions

Options for updating the label. The formatted text is used to update the label text, or select a background pixmap.

Enumerator:

UPDATE_TEXT Update the label text.

UPDATE_PIXMAP Update the label background pixmap.

9.67.3 Constructor & Destructor Documentation

9.67.3.1 QELabel::QELabel (QWidget * *parent* = 0)

Create without a variable. Use setVariableNameProperty() and setSubstitutions-Property() to define a variable and, optionally, macro substitutions later.

9.67.3.2 QELabel::QELabel (const QString & *variableName*, QWidget * *parent* = 0)

Create with a variable. A connection is automatically established. If macro substitutions are required, create without a variable and set the variable and macro substitutions after creation.

9.67.4 Member Function Documentation

9.67.4.1 void QELabel::dbValueChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.67.4.2 void QELabel::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.67.5 Property Documentation

9.67.5.1 bool QELabel::addUnits [read, write]

If true (default), add engineering units supplied with the data.

9.67.5.2 bool QELabel::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.67.5.3 ArrayActions QELabel::arrayAction [read, write]

Text formatting option for array data. Default is ASCII. Options are:

- ASCII - treat array as a single text string. For example an array of three characters 'a' 'b' 'c' will be formatted as 'abc'.
- APPEND - treat array as an array of numbers and format a string containing them all with a space between each. For example, an array of three numbers 10, 11 and 12 will be formatted as '10 11 12'.
- INDEX - Extract a single item from the array. The item is then formatted as any other non array data would be. The item selected is determined by the arrayIndex property. For example, if arrayIndex property is 1, an array of three numbers 10, 11 and 12 will be formatted as '11'.

9.67.5.4 bool QELabel::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is

invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.67.5.5 Formats QELabel::format [read, write]

Format to apply to data. Default is 'Default' in which case the data type supplied with the data determines how the data is formatted. For all other options, an attempt is made to format the data as requested (whatever its native form).

9.67.5.6 unsigned QELabel::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

Base used for when formatting integers. Default is 10 (duh!)

Index used to select a single item of data for formatting from an array of data. Default is 0. Only used when the arrayAction property is INDEX. Refer to the arrayAction property for more details.

9.67.5.7 bool QELabel::leadingZero [read, write]

If true (default), always add a leading zero when formatting numbers.

9.67.5.8 QString QELabel::localEnumeration [read, write]

An enumeration list used to data values. Used only when the formatting option is 'local enumeration'. Value is converted to an integer and used to select a string from this list.

9.67.5.9 Notations QELabel::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.67.5.10 QPixmap QELabel::pixmap0 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 0.

9.67.5.11 QPixmap QELabel::pixmap1 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 1.

9.67.5.12 QPixmap QELabel::pixmap2 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 2.

9.67.5.13 QPixmap QELabel::pixmap3 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 3.

9.67.5.14 QPixmap QELabel::pixmap4 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 4.

9.67.5.15 QPixmap QELabel::pixmap5 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 5.

9.67.5.16 QPixmap QELabel::pixmap6 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 6.

9.67.5.17 QPixmap QELabel::pixmap7 [read, write]

Pixmap displayed when updateOption property is 'Picture' and data is interpreted as 7.

9.67.5.18 int QELabel::precision [read, write]

Precision used when formatting floating point numbers. The default is 4. This is only used if useDbPrecision is false.

9.67.5.19 bool QELabel::trailingZeros [read, write]

If true (default), always remove any trailing zeros when formatting numbers.

9.67.5.20 UpdateOptions QELabel::updateOption [read, write]

Determines if data updates the label text, or the label pixmap. For both options all normal string formatting is applied. If Text, the formatted text is simply presented as the label text. If Picture, the FORMATTED text is then interpreted as an integer and used to select one of the pixmaps specified by properties pixmap0 through to pixmap7.

9.67.5.21 bool QELabel::useDbPrecision [read, write]

If true (default), format floating point numbers using the precision supplied with the data. If false, the precision property is used.

9.67.5.22 UserLevels QELabel::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through set-UserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.67.5.23 QString QELabel::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the style-Manager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.67.5.24 QString QELabel::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the style-Manager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.67.5.25 QString QELabel::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string

will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.67.5.26 UserLevels QELabel::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.67.5.27 QString QELabel::variable [read, write]

EPICS variable name (CA PV)

9.67.5.28 bool QELabel::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.67.5.29 QString QELabel::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.67.5.30 bool QELabel::visible [read, write]

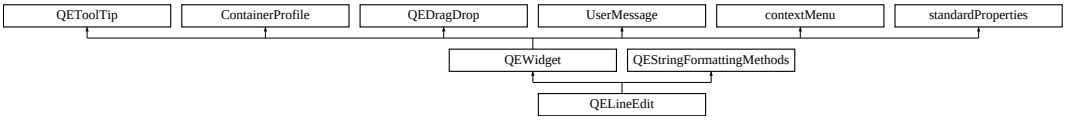
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QELabel/QELabel.h
- /home/andrew/epicsqt/framework/widgets/QELabel/QELabel.cpp

9.68 QELineEdit Class Reference

Inheritance diagram for QELineEdit:



Public Types

- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

- enum Formats { **Default** = QEStringFormatting::FORMAT_DEFAULT, **Floating** = QEStringFormatting::FORMAT_FLOATING, **Integer** = QEStringFormatting::FORMAT_INTEGER, **UnsignedInteger** = QEStringFormatting::FORMAT_UNSIGNEDINTEGER, **Time** = QEStringFormatting::FORMAT_TIME, **LocalEnumeration** = QEStringFormatting::FORMAT_LOCAL_ENUMERATE }

  User friendly enumerations for format property - refer to QEStringFormatting::formats for details.

- enum Notations { **Fixed** = QEStringFormatting::NOTATION_FIXED, **Scientific** = QEStringFormatting::NOTATION_SCIENTIFIC, **Automatic** = QEStringFormatting::NOTATION_AUTOMATIC }

  User friendly enumerations for notation property - refer to QEStringFormatting::notations for details.

- enum ArrayActions { **Append** = QEStringFormatting::APPEND, **Ascii** = QEStringFormatting::ASCII, **Index** = QEStringFormatting::INDEX }

  User friendly enumerations for arrayAction property - refer to QEStringFormatting::arrayActions for details.

Public Slots

- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const QString &out)
- void userChange (const QString &oldValue, const QString &newValue, const QString &lastValue)

  Internal use only. Used by QEConfiguredLayout to be notified when one of its widgets has written something.

- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.

Public Member Functions

- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setFormatProperty (Formats format)

  Access function for 'format' property - refer to 'format' property for details.
- Formats getFormatProperty ()

  Access function for 'format' property - refer to 'format' property for details.
- void setNotationProperty (Notations notation)

  Access function for 'notation' property - refer to 'notation' property for details.
- Notations getNotationProperty ()

  Access function for 'notation' property - refer to 'notation' property for details.
- void setArrayActionProperty (ArrayActions arrayAction)

  Access function for 'arrayAction' property - refer to 'arrayAction' property for details.
- ArrayActions getArrayActionProperty ()

  Access function for 'arrayAction' property - refer to 'arrayAction' property for details.
- QELineEdit (QWidget ∗parent=0)
- QELineEdit (const QString &variableName, QWidget ∗parent=0)
- void setWriteOnLoseFocus (bool writeOnLoseFocus)
- bool getWriteOnLoseFocus ()
- void setWriteOnEnter (bool writeOnEnter)
- bool getWriteOnEnter ()
- void setWriteOnFinish (bool writeOnFinish)
- bool getWriteOnFinish ()
- void setConfirmWrite (bool confirmWrite)
- bool getConfirmWrite ()
- void setSubscribe (bool subscribe)
- bool getSubscribe ()

Properties

- QString variable
- QString variableSubstitutions
- bool subscribe
- bool writeOnLoseFocus
- bool writeOnEnter
- bool writeOnFinish
- bool confirmWrite
- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- int precision
- bool useDbPrecision
- bool leadingZero
- bool trailingZeros
- bool addUnits
- QString localEnumeration
- Formats format
- Notations notation
- ArrayActions arrayAction

9.68.1 Constructor & Destructor Documentation

9.68.1.1 QELineEdit::QELineEdit (QWidget * *parent* = 0)

Create without a variable. Use setVariableNameProperty() and setSubstitutions-Property() to define a variable and, optionally, macro substitutions later.

9.68.1.2 QELineEdit::QELineEdit (const QString & *variableName,* QWidget * *parent* = 0)

Create with a variable. A connection is automatically established. If macro substitutions are required, create without a variable and set the variable and macro substitutions after creation.

9.68.2 Member Function Documentation

9.68.2.1 void QELineEdit::dbValueChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.68.2.2 bool QELineEdit::getConfirmWrite ( )

Returns 'true' if this widget will ask for confirmation (using a dialog box) prior to writing data.

9.68.2.3 bool QELineEdit::getSubscribe ( )

Returns 'true' if this widget subscribes for data updates and displays current data.

9.68.2.4 bool QELineEdit::getWriteOnEnter ( )

Returns 'true' if this widget writes any changes when the user presses 'enter'.

9.68.2.5 bool QELineEdit::getWriteOnFinish ( )

Returns 'true' if this widget writes any changes when the user finished editing (the QLineEdit 'editingFinished' signal is emitted).

9.68.2.6 bool QELineEdit::getWriteOnLoseFocus ( )

Returns 'true' if this widget automatically writes any changes when it loses focus.

9.68.2.7 void QELineEdit::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.68.2.8 void QELineEdit::setConfirmWrite (bool *confirmWrite*)

Sets if this widget will ask for confirmation (using a dialog box) prior to writing data. Default is 'false' (will not ask for confirmation (using a dialog box) prior to writing data).

9.68.2.9 void QELineEdit::setSubscribe (bool *subscribe*)

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

9.68.2.10 void QELineEdit::setWriteOnEnter (bool *writeOnEnter*)

Sets if this widget writes any changes when the user presses 'enter'. Note, the current value will be written even if the user has not changed it. Default is 'true' (writes any changes when the user presses 'enter').

9.68.2.11 void QELineEdit::setWriteOnFinish (bool *writeOnFinish*)

Sets if this widget writes any changes when the user finished editing (the QLineEdit 'editingFinished' signal is emitted). No writing occurs if no changes were made. Default is 'true' (writes any changes when the QLineEdit 'editingFinished' signal is emitted).

9.68.2.12 void QELineEdit::setWriteOnLoseFocus (bool *writeOnLoseFocus*)

Sets if this widget automatically writes any changes when it loses focus. Default is 'false' (does not write any changes when it loses focus).

9.68.3 Property Documentation

9.68.3.1 bool QELineEdit::addUnits [read, write]

If true (default), add engineering units supplied with the data.

9.68.3.2 bool QELineEdit::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.68.3.3 ArrayActions QELineEdit::arrayAction [read, write]

Text formatting option for array data. Default is ASCII. Options are:

- ASCII - treat array as a single text string. For example an array of three characters 'a' 'b' 'c' will be formatted as 'abc'.
- APPEND - treat array as an array of numbers and format a string containing them all with a space between each. For example, an array of three numbers 10, 11 and 12 will be formatted as '10 11 12'.

- INDEX - Extract a single item from the array. The item is then formatted as any other non array data would be. The item selected is determined by the arrayIndex property. For example, if arrayIndex property is 1, an array of three numbers 10, 11 and 12 will be formatted as '11'.

9.68.3.4 bool QELineEdit::confirmWrite [read, write]

Sets if this widget will ask for confirmation (using a dialog box) prior to writing data. Default is 'false' (will not ask for confirmation (using a dialog box) prior to writing data).

9.68.3.5 bool QELineEdit::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.68.3.6 Formats QELineEdit::format [read, write]

Format to apply to data. Default is 'Default' in which case the data type supplied with the data determines how the data is formatted. For all other options, an attempt is made to format the data as requested (whatever its native form).

9.68.3.7 unsigned QELineEdit::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

Base used for when formatting integers. Default is 10 (duh!)

Index used to select a single item of data for formatting from an array of data. Default is 0. Only used when the arrayAction property is INDEX. Refer to the arrayAction property for more details.

9.68.3.8 bool QELineEdit::leadingZero [read, write]

If true (default), always add a leading zero when formatting numbers.

9.68.3.9 QString QELineEdit::localEnumeration [read, write]

An enumeration list used to data values. Used only when the formatting option is 'local enumeration'. Value is converted to an integer and used to select a string from this list.

9.68.3.10 Notations QELineEdit::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.68.3.11 int QELineEdit::precision [read, write]

Precision used when formatting floating point numbers. The default is 4. This is only used if useDbPrecision is false.

9.68.3.12 bool QELineEdit::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.68.3.13 bool QELineEdit::trailingZeros [read, write]

If true (default), always remove any trailing zeros when formatting numbers.

9.68.3.14 bool QELineEdit::useDbPrecision [read, write]

If true (default), format floating point numbers using the precision supplied with the data. If false, the precision property is used.

9.68.3.15 UserLevels QELineEdit::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.68.3.16 QString QELineEdit::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.68.3.17 QString QELineEdit::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the style-Manager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.68.3.18 QString QELineEdit::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.68.3.19 UserLevels QELineEdit::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.68.3.20 QString QELineEdit::variable [read, write]

EPICS variable name (CA PV)

9.68.3.21 bool QELineEdit::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.68.3.22 QString QELineEdit::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.68.3.23 bool QELineEdit::visible [read, write]

Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

9.68.3.24 bool QELineEdit::writeOnEnter [read, write]

Sets if this widget writes any changes when the user presses 'enter'. Note, the current value will be written even if the user has not changed it. Default is 'true' (writes any changes when the user presses 'enter').

9.68.3.25 bool QELineEdit::writeOnFinish [read, write]

Sets if this widget writes any changes when the user finished editing (the QLineEdit 'editingFinished' signal is emitted). No writing occurs if no changes were made. Default is 'true' (writes any changes when the QLineEdit 'editingFinished' signal is emitted).

9.68.3.26 bool QELineEdit::writeOnLoseFocus [read, write]

Sets if this widget automatically writes any changes when it loses focus. Default is 'false' (does not write any changes when it loses focus).

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QELineEdit/QELineEdit.h
- /home/andrew/epicsqt/framework/widgets/QELineEdit/QELineEdit.cpp

9.69 QELineEditManager Class Reference

Public Member Functions

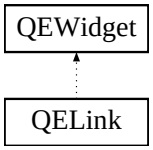
- **QELineEditManager** (QObject *parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget * **createWidget** (QWidget *parent)
- void **initialize** (QDesignerFormEditorInterface *core)

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QELineEdit/QELineEditManager.h

9.70 QELink Class Reference

Inheritance diagram for QELink:



Public Types

- enum **conditions** { **CONDITION_EQ**, **CONDITION_NE**, **CONDITION_GT**, **CONDITION_GE**, **CONDITION_LT**, **CONDITION_LE** }
- enum **ConditionNames** { **Equal** = QELink::CONDITION_EQ, **NotEqual** = QELink::CONDITION_NE, **GreaterThan** = QELink::CONDITION_GT, **GreaterThanOrEqual** = QELink::CONDITION_GE, **LessThan** = QELink::CONDITION_LT, **LessThanOrEqual** = QELink::CONDITION_LE }

Public Slots

- void **in** (const bool &in)
- void **in** (const qlonglong &in)
- void **in** (const double &in)
- void **in** (const QString &in)
- void **autoFillBackground** (const bool &enable)

Signals

- void **out** (const bool &out)
- void **out** (const qlonglong &out)
- void **out** (const double &out)
- void **out** (const QString &out)

Public Member Functions

- **QELink** (QWidget ∗parent=0)
- void **setCondition** (conditions conditionIn)
- conditions **getCondition** ()
- void **setComparisonValue** (QString comparisonValue)
- QString **getComparisonValue** ()

- void **setSignalTrue** (bool signalTrue)
- bool **getSignalTrue** ()
- void **setSignalFalse** (bool signalFalse)
- bool **getSignalFalse** ()
- void **setOutTrueValue** (QString outTrueValue)
- QString **getOutTrueValue** ()
- void **setOutFalseValue** (QString outFalseValue)
- QString **getOutFalseValue** ()
- void **setRunVisible** (bool visibleIn)
- bool **getRunVisible** ()
- void **setConditionProperty** (ConditionNames condition)
- ConditionNames **getConditionProperty** ()

Protected Attributes

- conditions **condition**
- QVariant **comparisonValue**
- bool **signalTrue**
- bool **signalFalse**
- QVariant **outTrueValue**
- QVariant **outFalseValue**
- bool **visible**

Properties

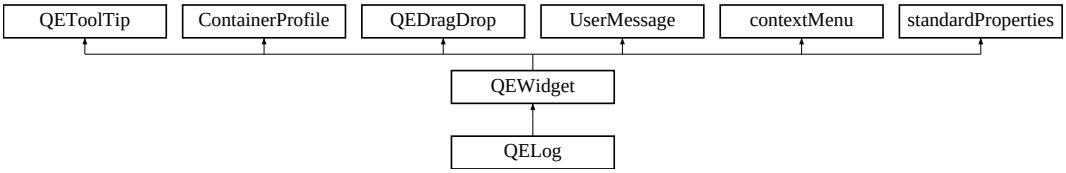
- ConditionNames **condition**
- QString **comparisonValue**
- QString **outTrueValue**
- QString **outFalseValue**
- bool **runVisible**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QELink/QELink.h
- /home/andrew/epicsqt/framework/widgets/QELink/QELink.cpp

9.71 QELog Class Reference

Inheritance diagram for QELog:



Public Types

- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **MessageFilterOptions** { **Any** = UserMessage::MESSAGE_FILTER_ANY, **Match** = UserMessage::MESSAGE_FILTER_MATCH, **None** = UserMessage::-MESSAGE_FILTER_NONE }

Public Member Functions

- **QELog** (QWidget ∗pParent=0)
- void **setShowColumnTime** (bool pValue)
- bool **getShowColumnTime** ()
- void **setShowColumnType** (bool pValue)
- bool **getShowColumnType** ()
- void **setShowColumnMessage** (bool pValue)
- bool **getShowColumnMessage** ()
- void **setShowMessageFilter** (bool pValue)
- bool **getShowMessageFilter** ()
- void **setShowClear** (bool pValue)
- bool **getShowClear** ()
- void **setShowSave** (bool pValue)
- bool **getShowSave** ()
- void **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- void **setScrollToBottom** (bool pValue)
- bool **getScrollToBottom** ()
- void **setInfoColor** (QColor pValue)
- QColor **getInfoColor** ()
- void **setWarningColor** (QColor pValue)
- QColor **getWarningColor** ()
- void **setErrorColor** (QColor pValue)
- QColor **getErrorColor** ()
- void **clearLog** ()
- void **addLog** (int pType, QString pMessage)
- void **refreshLog** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()
- MessageFilterOptions **getMessageFormFilter** ()
- void **setMessageFormFilter** (MessageFilterOptions messageFormFilter)
- MessageFilterOptions **getMessageSourceFilter** ()
- void **setMessageSourceFilter** (MessageFilterOptions messageSourceFilter)

Protected Attributes

- _QTableWidgetLog * **qTableWidgetLog**
- QCheckBox * **qCheckBoxInfoMessage**
- QCheckBox * **qCheckBoxWarningMessage**
- QCheckBox * **qCheckBoxErrorMessage**
- QPushButton * **qPushButtonClear**
- QPushButton * **qPushButtonSave**
- QColor **qColorInfo**
- QColor **qColorWarning**
- QColor **qColorError**
- bool **scrollToBottom**
- int **detailsLayout**

Properties

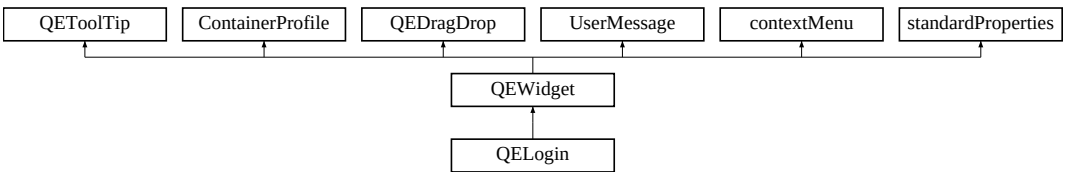
- bool **showColumnTime**
- bool **showColumnType**
- bool **showColumnMessage**
- bool **showMessageFilter**
- bool **showClear**
- bool **showSave**
- detailsLayoutProperty **detailsLayout**
- QColor **infoColor**
- QColor **warningColor**
- QColor **errorColor**
- MessageFilterOptions **messageFormFilter**
- MessageFilterOptions **messageSourceFilter**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QELog/QELog.h
- /home/andrew/epicsqt/framework/widgets/QELog/QELog.cpp

9.72 QELogin Class Reference

Inheritance diagram for QELogin:



Public Types

- enum **userTypesProperty** { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }
- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }

Public Member Functions

- **QELogin** (QWidget *pParent=0)
- void **setShowUserType** (bool pValue)
- bool **getShowUserType** ()
- void **setShowLogin** (bool pValue)
- bool **getShowButtonLogin** ()
- void **setShowLogout** (bool pValue)
- bool **getShowButtonLogout** ()
- void **setUserPassword** (QString pValue)
- QString **getUserPassword** ()
- void **setScientistPassword** (QString pValue)
- QString **getScientistPassword** ()
- void **setEngineerPassword** (QString pValue)
- QString **getEngineerPassword** ()
- void **setCurrentUserType** (int pValue)
- int **getCurrentUserType** ()
- void **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- QString **getUserTypeName** (userLevels type)
- void **logoutCurrentUserType** ()
- void **setCurrentUserTypeProperty** (userTypesProperty pUserType)
- userTypesProperty **getCurrentUserTypeProperty** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()

Protected Attributes

- QStack< int > **loginHistory**
- QPushButton * **qPushButtonLogin**
- QPushButton * **qPushButtonLogout**
- QLabel * **qLabelUserType**
- QString **userPassword**
- QString **scientistPassword**
- QString **engineerPassword**
- int **currentUserType**
- int **detailsLayout**

Properties

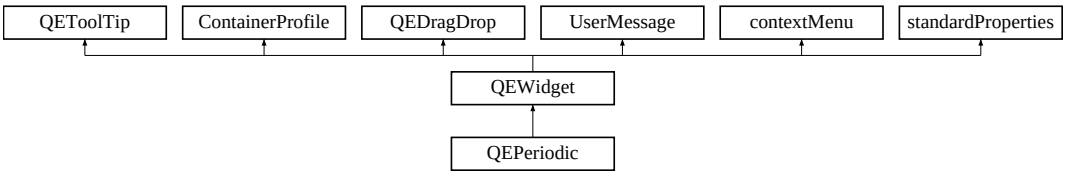
- bool **showUserType**
- bool **showLogin**
- bool **showLogout**
- userTypesProperty **currentUserType**
- detailsLayoutProperty **detailsLayout**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QELogin/QELogin.h
- /home/andrew/epicsqt/framework/widgets/QELogin/QELogin.cpp

9.73 QEPeriodic Class Reference

Inheritance diagram for QEPeriodic:



Classes

- struct elementInfoStruct
- struct userInfoStructArray

Public Types

- enum **variableTypes** { **VARIABLE_TYPE_NUMBER**, **VARIABLE_TYPE_ATOMIC_WEIGHT**, **VARIABLE_TYPE_MELTING_POINT**, **VARIABLE_TYPE_BOILING_POINT**, **VARIABLE_TYPE_DENSITY**, **VARIABLE_TYPE_GROUP**, **VARIABLE_TYPE_IONIZATION_ENERGY**, **VARIABLE_TYPE_USER_VALUE_1**, **VARIABLE_TYPE_USER_VALUE_2** }
- enum **presentationOptions** { **PRESENTATION_BUTTON_AND_LABEL**, **PRESENTATION_BUTTON_ONLY**, **PRESENTATION_LABEL_ONLY** }
- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.
- enum **PresentationOptions** { **buttonAndLabel** = QEPeriodic::PRESENTATION_BUTTON_AND_LABEL, **buttonOnly** = QEPeriodic::PRESENTATION_BUTTON_ONLY, **labelOnly** = QEPeriodic::PRESENTATION_LABEL_ONLY }

- enum **VariableTypes** { **Number** = QEPeriodic::VARIABLE_TYPE_NUMBER, **atomicWeight** = QEPeriodic::VARIABLE_TYPE_ATOMIC_WEIGHT, **meltingPoint** = QEPeriodic::VARIABLE_TYPE_MELTING_POINT, **boilingPoint** = QEPeriodic::VARIABLE_TYPE_BOILING_POINT, **density** = QEPeriodic::VARIABLE_TYPE_DENSITY, **group** = QEPeriodic::VARIABLE_TYPE_GROUP, **ionizationEnergy** = QEPeriodic::VARIABLE_TYPE_IONIZATION_ENERGY, **userValue1** = QEPeriodic::VARIABLE_TYPE_USER_VALUE_1, **userValue2** = QEPeriodic::VARIABLE_TYPE_USER_VALUE_2 }

Public Slots

- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const double &out)
- void dbElementChanged (const QString &out)
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.

Public Member Functions

- **QEPeriodic** (QWidget *parent=0)
- **QEPeriodic** (const QString &variableName, QWidget *parent=0)
- void **setSubscribe** (bool subscribe)
- bool **getSubscribe** ()
- void **setPresentationOption** (presentationOptions presentationOptionIn)
- presentationOptions **getPresentationOption** ()
- void **setVariableType1** (variableTypes variableType1In)
- variableTypes **getVariableType1** ()
- void **setVariableType2** (variableTypes variableType2In)
- variableTypes **getVariableType2** ()
- void **setVariableTolerance1** (double variableTolerance1In)
- double **getVariableTolerance1** ()
- void **setVariableTolerance2** (double variableTolerance2In)
- double **getVariableTolerance2** ()
- void **setUserInfo** (QString userInfo)
- QString **getUserInfo** ()
- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.

- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.

- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.

- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.

- void **setPresentationOptionProperty** (PresentationOptions presentationOption)
- PresentationOptions **getPresentationOptionProperty** ()
- void **setVariableType1Property** (VariableTypes variableType)
- void **setVariableType2Property** (VariableTypes variableType)
- VariableTypes **getVariableType1Property** ()
- VariableTypes **getVariableType2Property** ()

Public Attributes

- userInfoStruct **userInfo** [NUM_ELEMENTS]

Static Public Attributes

- static elementInfoStruct **elementInfo** [NUM_ELEMENTS]

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent *event)
- void **dropEvent** (QDropEvent *event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Protected Attributes

- QEFloatingFormatting **floatingFormatting**
- bool **localEnabled**
- bool allowDrop
- variableTypes **variableType1**
- variableTypes **variableType2**
- double **variableTolerance1**
- double **variableTolerance2**

Properties

- QString writeButtonVariable1
- QString writeButtonVariable2
- QString readbackLabelVariable1
- QString readbackLabelVariable2
- QString variableSubstitutions
- bool subscribe
- bool variableAsToolTip
- bool enabled
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- PresentationOptions **presentationOption**
- VariableTypes **variableType1**
- VariableTypes **variableType2**
- QString **userInfo**

9.73.1 Member Function Documentation

9.73.1.1 void QEPeriodic::dbElementChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.73.1.2 void QEPeriodic::dbValueChanged (const double & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.73.1.3 void QEPeriodic::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.73.2 Member Data Documentation

9.73.2.1 bool QEPeriodic::allowDrop [read, write, protected]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.73.3 Property Documentation

9.73.3.1 bool QEPeriodic::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.73.3.2 unsigned QEPeriodic::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.73.3.3 QString QEPeriodic::readbackLabelVariable1 [read, write]

EPICS variable name (CA PV). This variable is used to read the value to the first of two positioners to determine which (if any) element is currently selected.

9.73.3.4 QString QEPeriodic::readbackLabelVariable2 [read, write]

EPICS variable name (CA PV). This variable is used to read the value to the second of two positioners to determine which (if any) element is currently selected.

9.73.3.5 bool QEPeriodic::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.73.3.6 UserLevels QEPeriodic::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.73.3.7 QString QEPeriodic::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.73.3.8 QString QEPeriodic::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.73.3.9 QString QEPeriodic::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.73.3.10 UserLevels QEPeriodic::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.73.3.11 bool QEPeriodic::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.73.3.12 QString QEPeriodic::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'SAMPLE=SAM1, NAME = "Ref foil"' These substitutions are applied to all the variable names.

9.73.3.13 bool QEPeriodic::visible [read, write]

Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

9.73.3.14 QString QEPeriodic::writeButtonVariable1 [read, write]

EPICS variable name (CA PV). This variable is used to write a value to the first of two positioners that will position the select element.

9.73.3.15 QString QEPeriodic::writeButtonVariable2 [read, write]

EPICS variable name (CA PV). This variable is used to write a value to the second of two positioners that will position the select element.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEPeriodic/QEPeriodic.h
- /home/andrew/epicsqt/framework/widgets/QEPeriodic/QEPeriodic.cpp

9.74 QEPeriodicComponentData Class Reference

Public Attributes

- unsigned int **variableIndex1**
- double **lastData1**
- bool **haveLastData1**
- unsigned int **variableIndex2**
- double **lastData2**
- bool **haveLastData2**

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEPeriodic/QEPeriodic.h

9.75 QEPeriodicTaskMenu Class Reference

Public Member Functions

- **QEPeriodicTaskMenu** (QEPeriodic ∗periodic, QObject ∗parent)
- QAction ∗ **preferredEditAction** () const
- QList< QAction ∗ > **taskActions** () const

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEPeriodic/QEPeriodicTaskMenu.h
- /home/andrew/epicsqt/framework/widgets/QEPeriodic/QEPeriodicTaskMenu-Extension.cpp

9.76 QEPeriodicTaskMenuFactory Class Reference

Public Member Functions

- **QEPeriodicTaskMenuFactory** (QExtensionManager ∗parent=0)

Protected Member Functions

- QObject ∗ **createExtension** (QObject ∗object, const QString &iid, QObject ∗parent) const

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEPeriodic/QEPeriodicTaskMenu.h
- /home/andrew/epicsqt/framework/widgets/QEPeriodic/QEPeriodicTaskMenu-Extension.cpp

9.77 QEpicsPV Class Reference

Public Slots

- const QVariant & **set** (QVariant value, int delay=-1)
- void **setPV** (const QString &_pvName="")

Signals

- void **connectionChanged** (bool connected)
- void **connected** ()
- void **disconnected** ()
- void **valueChanged** (const QVariant &value)
- void **valueUpdated** (const QVariant &value)
- void **valueInited** (const QVariant &value)

Public Member Functions

- **QEpicsPV** (const QString &_pvName, QObject ∗parent=0)
- **QEpicsPV** (QObject ∗parent=0)
- const QVariant & **get** () const
- void **needUpdated** () const
- const QVariant & **getUpdated** (int delay=defaultDelay) const
- bool **isConnected** () const
- const QStringList & **getEnum** () const
- const QString & **pv** () const
- const QVariant & **getReady** (int delay=defaultDelay) const

Static Public Member Functions

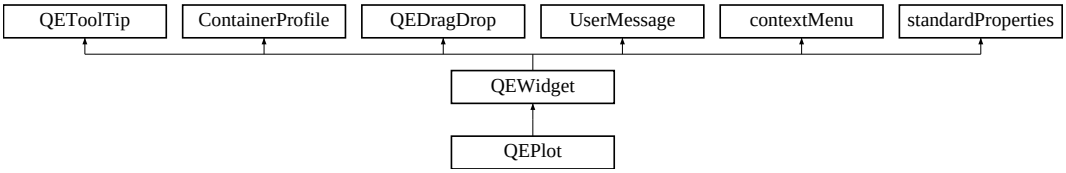
- static void **setDebugLevel** (unsigned level=0)
- static QVariant **get** (const QString &_pvName, int delay=defaultDelay)
- static QVariant **set** (QString &_pvName, const QVariant &value, int delay=-1)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/qepicspv.h
- /home/andrew/epicsqt/framework/data/src/qepicspv.cpp

9.78 QEPlot Class Reference

Inheritance diagram for QEPlot:



Public Types

- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.
- enum **TraceStyles** { **Lines** = QwtPlotCurve::Lines, **Sticks** = QwtPlotCurve::Sticks, **Steps** = QwtPlotCurve::Steps, **Dots** = QwtPlotCurve::Dots }

Public Slots

- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const double &out)
- void dbValueChanged (const QVector< double > &out)

Public Member Functions

- **QEPlot** (QWidget *parent=0)
- **QEPlot** (const QString &variableName, QWidget *parent=0)
- void **setYMin** (double yMin)
- double **getYMin** ()
- void **setYMax** (double yMax)
- double **getYMax** ()
- void **setAutoScale** (bool autoScale)
- bool **getAutoScale** ()
- void **setAxisEnableX** (bool axisEnableXIn)
- bool **getAxisEnableX** ()
- void **setAxisEnableY** (bool axisEnableYIn)
- bool **getAxisEnableY** ()
- QString **getTitle** ()
- void **setBackgroundColor** (QColor backgroundColor)
- QColor **getBackgroundColor** ()
- void **setTraceStyle** (QwtPlotCurve::CurveStyle traceStyle, const unsigned int variableIndex)
- QwtPlotCurve::CurveStyle **getTraceStyle** (const unsigned int variableIndex)
- void **setTraceColor** (QColor traceColor, const unsigned int variableIndex)
- void **setTraceColor1** (QColor traceColor)
- void **setTraceColor2** (QColor traceColor)
- void **setTraceColor3** (QColor traceColor)
- void **setTraceColor4** (QColor traceColor)
- QColor **getTraceColor** (const unsigned int variableIndex)

- QColor **getTraceColor1** ()
- QColor **getTraceColor2** ()
- QColor **getTraceColor3** ()
- QColor **getTraceColor4** ()
- void **setTraceLegend1** (QString traceLegend)
- void **setTraceLegend2** (QString traceLegend)
- void **setTraceLegend3** (QString traceLegend)
- void **setTraceLegend4** (QString traceLegend)
- QString **getTraceLegend1** ()
- QString **getTraceLegend2** ()
- QString **getTraceLegend3** ()
- QString **getTraceLegend4** ()
- void **setXUnit** (QString xUnit)
- QString **getXUnit** ()
- void **setYUnit** (QString yUnit)
- QString **getYUnit** ()
- void **setGridEnableMajorX** (bool gridEnableMajorXIn)
- void **setGridEnableMajorY** (bool gridEnableMajorYIn)
- void **setGridEnableMinorX** (bool gridEnableMinorXIn)
- void **setGridEnableMinorY** (bool gridEnableMinorYIn)
- bool **getGridEnableMajorX** ()
- bool **getGridEnableMajorY** ()
- bool **getGridEnableMinorX** ()
- bool **getGridEnableMinorY** ()
- void **setGridMajorColor** (QColor gridMajorColorIn)
- void **setGridMinorColor** (QColor gridMinorColorIn)
- QColor **getGridMajorColor** ()
- QColor **getGridMinorColor** ()
- void **setXStart** (double xStart)
- double **getXStart** ()
- void **setXIncrement** (double xIncrement)
- double **getXIncrement** ()
- void **setTimeSpan** (unsigned int timeSpan)
- unsigned int **getTimeSpan** ()
- void **setTickRate** (unsigned int tickRate)
- unsigned int **getTickRate** ()
- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.

- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void **setTraceStyle1** (TraceStyles traceStyle)
- void **setTraceStyle2** (TraceStyles traceStyle)
- void **setTraceStyle3** (TraceStyles traceStyle)
- void **setTraceStyle4** (TraceStyles traceStyle)
- TraceStyles **getTraceStyle1** ()
- TraceStyles **getTraceStyle2** ()
- TraceStyles **getTraceStyle3** ()
- TraceStyles **getTraceStyle4** ()

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **mousePressEvent** (QMouseEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Protected Attributes

- QEFloatingFormatting **floatingFormatting**
- bool **localEnabled**
- bool allowDrop

Properties

- QString variable1
- QString variable2
- QString variable3
- QString variable4
- QString variableSubstitutions
- bool variableAsToolTip
- bool enabled
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility

- UserLevels userLevelEnabled
- QColor **traceColor1**
- QColor **traceColor2**
- QColor **traceColor3**
- QColor **traceColor4**
- TraceStyles **traceStyle1**
- TraceStyles **traceStyle2**
- TraceStyles **traceStyle3**
- TraceStyles **traceStyle4**
- QString **traceLegend1**
- QString **traceLegend2**
- QString **traceLegend3**
- QString **traceLegend4**
- QString **title**
- QColor **backgroundColor**
- QString **xUnit**
- QString **yUnit**

9.78.1 Member Function Documentation

9.78.1.1 void QEPlot::dbValueChanged (const double & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.78.1.2 void QEPlot::dbValueChanged (const QVector< double > & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.78.1.3 void QEPlot::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.78.2 Member Data Documentation

9.78.2.1 bool QEPlot::allowDrop [read, write, protected]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.78.3 Property Documentation

9.78.3.1 bool QEPlot::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.78.3.2 unsigned QEPlot::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.78.3.3 UserLevels QEPlot::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.78.3.4 QString QEPlot::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.78.3.5 QString QEPlot::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.78.3.6 QString QEPlot::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.78.3.7 UserLevels QEPlot::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.78.3.8 QString QEPlot::variable1 [read, write]

EPICS variable name (CA PV). This variable is used to read updating values or waveforms for plotting in the first trace.

9.78.3.9 QString QEPlot::variable2 [read, write]

EPICS variable name (CA PV). This variable is used to read updating values or waveforms for plotting in the second trace.

9.78.3.10 QString QEPlot::variable3 [read, write]

EPICS variable name (CA PV). This variable is used to read updating values or waveforms for plotting in the third trace.

9.78.3.11 QString QEPlot::variable4 [read, write]

EPICS variable name (CA PV). This variable is used to read updating values or waveforms for plotting in the fourth trace.

9.78.3.12 bool QEPlot::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.78.3.13 QString QEPlot::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'SAMPLE=SAM1, NAME = "Ref foil"' These substitutions are applied to all the variable names.

9.78.3.14 bool QEPlot::visible [read, write]

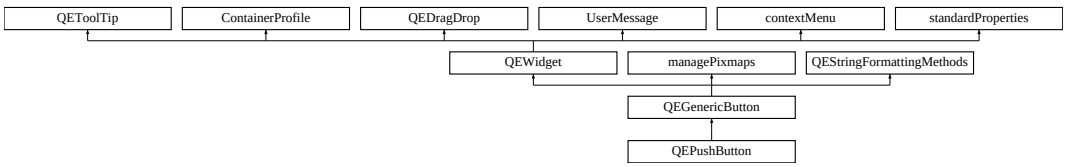
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEPlot/QEPlot.h
- /home/andrew/epicsqt/framework/widgets/QEPlot/QEPlot.cpp

9.79 QEPushButton Class Reference

Inheritance diagram for QEPushButton:



Public Types

- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.
- enum **UpdateOptions** { **Text** = QEPushButton::UPDATE_TEXT, **Icon** = QEPushButton::UPDATE_ICON, **TextAndIcon** = QEPushButton::UPDATE_TEXT_AND_ICON, **State** = QEPushButton::UPDATE_STATE }
- enum **Formats** { **Default** = QEStringFormatting::FORMAT_DEFAULT, **Floating** = QEStringFormatting::FORMAT_FLOATING, **Integer** = QEStringFormatting::FORMAT_INTEGER, **UnsignedInteger** = QEStringFormatting::FORMAT_UNSIGNEDINTEGER, **Time** = QEStringFormatting::FORMAT_TIME, **LocalEnumeration** = QEStringFormatting::FORMAT_LOCAL_ENUMERATE }
- enum **Notations** { **Fixed** = QEStringFormatting::NOTATION_FIXED, **Scientific** = QEStringFormatting::NOTATION_SCIENTIFIC, **Automatic** = QEStringFormatting::NOTATION_AUTOMATIC }

- enum **CreationOptionNames** { **Open** = QEForm::CREATION_OPTION_OPEN, **NewTab** = QEForm::CREATION_OPTION_NEW_TAB, **NewWindow** = QEForm::CREATION_OPTION_NEW_WINDOW }

Public Slots

- void **launchGui** (QString guiName, QEForm::creationOptions creationOption)
- void **onGeneralMessage** (QString message)
- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const QString &out)
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.
- void newGui (QString guiName, QEForm::creationOptions creationOption)

  Internal use only. Request a new GUI is created. Typically, this is caught by the QEGui application.

Public Member Functions

- **QEPushButton** (QWidget *parent=0)
- **QEPushButton** (const QString &variableName, QWidget *parent=0)
- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.

Protected Member Functions

- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Properties

- QString variable
- QString altReadbackVariable
- QString **variableSubstitutions**
- bool subscribe
- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- UpdateOptions **updateOption**
- QPixmap **pixmap0**
- QPixmap **pixmap1**
- QPixmap **pixmap2**
- QPixmap **pixmap3**
- QPixmap **pixmap4**
- QPixmap **pixmap5**
- QPixmap **pixmap6**
- QPixmap **pixmap7**
- bool **useDbPrecision**
- bool **leadingZero**
- bool **trailingZeros**
- bool **addUnits**
- QString **localEnumeration**
- Qt::Alignment **alignment**
- Formats **format**
- Notations **notation**
- QString **password**
- bool **confirmAction**
- bool **writeOnPress**
- bool **writeOnRelease**
- bool **writeOnClick**
- QString **pressText**

- QString **releaseText**
- QString **clickText**
- QString **clickCheckedText**
- QString **labelText**
- QString **program**
- QStringList **arguments**
- QString **guiFile**
- CreationOptionNames **creationOption**

9.79.1 Member Function Documentation

9.79.1.1 void QEPushButton::dbValueChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.79.1.2 void QEPushButton::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.79.2 Property Documentation

9.79.2.1 bool QEPushButton::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.79.2.2 QString QEPushButton::altReadbackVariable [read, write]

EPICS variable name (CA PV). This variable is used to provide a readback value when different to the variable written to by a button press.

9.79.2.3 bool QEPushButton::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.79.2.4 unsigned QEPushButton::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.79.2.5 bool QEPushButton::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.79.2.6 UserLevels QEPushButton::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.79.2.7 QString QEPushButton::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.79.2.8 QString QEPushButton::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.79.2.9 QString QEPushButton::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example,

'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.79.2.10 UserLevels QEPushButton::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.79.2.11 QString QEPushButton::variable [read, write]

EPICS variable name (CA PV). This variable is used for both writing (on button press), and reading if subscribed and no alternate readback variable is provided.

9.79.2.12 bool QEPushButton::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.79.2.13 bool QEPushButton::visible [read, write]

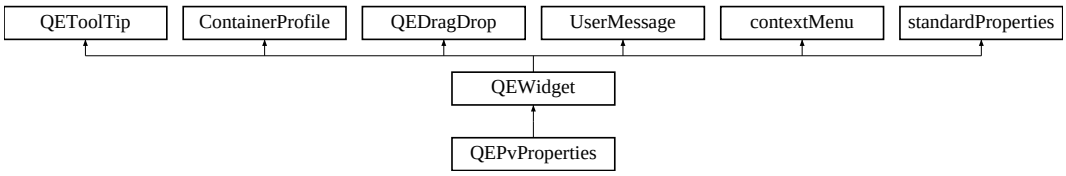
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEButton/QEPushButton.h
- /home/andrew/epicsqt/framework/widgets/QEButton/QEPushButton.cpp

9.80 QEPvProperties Class Reference

Inheritance diagram for QEPvProperties:



Classes

- struct WidgetHolder

Public Types

- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Public Slots

- void requestEnabled (const bool &state)

Signals

- void **setCurrentBoxIndex** (int index)

Public Member Functions

- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.

- **QEPvProperties** (QWidget *parent=0)
- **QEPvProperties** (const QString &variableName, QWidget *parent=0)
- QSize **sizeHint** () const
- void **establishConnection** (unsigned int variableIndex)
- void **updateToolTip** (const QString &tip)

Protected Member Functions

- void **setup** ()
- qcaobject::QCaObject * **createQcaItem** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent *event)
- void **dropEvent** (QDropEvent *event)
- void **mousePressEvent** (QMouseEvent *event)
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Properties

- QString variable
- QString variableSubstitutions
- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled

9.80.1 Member Function Documentation

9.80.1.1 void QEPvProperties::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.80.2 Property Documentation

9.80.2.1 bool QEPvProperties::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.80.2.2 bool QEPvProperties::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.80.2.3 unsigned QEPvProperties::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.80.2.4 UserLevels QEPvProperties::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.80.2.5 QString QEPvProperties::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.80.2.6 QString QEPvProperties::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the style-Manager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.80.2.7 QString QEPvProperties::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.80.2.8 UserLevels QEPvProperties::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.80.2.9 QString QEPvProperties::variable [read, write]

EPICS variable name (CA PV)

9.80.2.10 bool QEPvProperties::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.80.2.11 QString QEPvProperties::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.80.2.12 bool QEPvProperties::visible [read, write]

Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEPvProperties/QEPvProperties.h
- /home/andrew/epicsqt/framework/widgets/QEPvProperties/QEPvProperties.cpp

9.81 QEPvPropertiesManager Class Reference

Public Member Functions

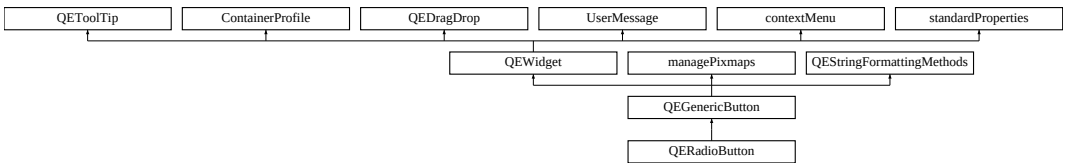
- **QEPvPropertiesManager** (QObject *parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget * **createWidget** (QWidget *parent)
- void **initialize** (QDesignerFormEditorInterface *core)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEPvProperties/QEPvProperties-Manager.h
- /home/andrew/epicsqt/framework/widgets/QEPvProperties/QEPvProperties-Manager.cpp

9.82 QERadioButton Class Reference

Inheritance diagram for QERadioButton:



Public Types

- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.
- enum **UpdateOptions** { **Text** = QERadioButton::UPDATE_TEXT, **Icon** = QERadioButton::UPDATE_ICON, **TextAndIcon** = QERadioButton::UPDATE_TEXT_AND_ICON, **State** = QERadioButton::UPDATE_STATE }
- enum **Formats** { **Default** = QEStringFormatting::FORMAT_DEFAULT, **Floating** = QEStringFormatting::FORMAT_FLOATING, **Integer** = QEStringFormatting::FORMAT_INTEGER, **UnsignedInteger** = QEStringFormatting::FORMAT_UNSIGNEDINTEGER, **Time** = QEStringFormatting::FORMAT_TIME }
- enum **Notations** { **Fixed** = QEStringFormatting::NOTATION_FIXED, **Scientific** = QEStringFormatting::NOTATION_SCIENTIFIC, **Automatic** = QEStringFormatting::NOTATION_AUTOMATIC }
- enum **CreationOptionNames** { **Open** = QEForm::CREATION_OPTION_OPEN, **NewTab** = QEForm::CREATION_OPTION_NEW_TAB, **NewWindow** = QEForm::CREATION_OPTION_NEW_WINDOW }

Public Slots

- void **launchGui** (QString guiName, QEForm::creationOptions creationOption)
- void **onGeneralMessage** (QString message)
- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const QString &out)
- void requestResend ()

  Internal use only. Used when changing a property value to force a re-display to reflect the new property value.
- void newGui (QString guiName, QEForm::creationOptions creationOption)

  Internal use only. Request a new GUI is created. Typically, this is caught by the QEGui application.

Public Member Functions

- **QERadioButton** (QWidget *parent=0)
- **QERadioButton** (const QString &variableName, QWidget *parent=0)
- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.

- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.

Protected Member Functions

- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Properties

- QString variable
- QString variableSubstitutions
- bool subscribe
- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- UpdateOptions **updateOption**
- QPixmap **pixmap0**
- QPixmap **pixmap1**
- QPixmap **pixmap2**
- QPixmap **pixmap3**
- QPixmap **pixmap4**
- QPixmap **pixmap5**
- QPixmap **pixmap6**
- QPixmap **pixmap7**
- bool **useDbPrecision**

- bool **leadingZero**
- bool **trailingZeros**
- bool **addUnits**
- Qt::Alignment **alignment**
- Formats **format**
- Notations **notation**
- QString **password**
- bool **confirmAction**
- bool **writeOnPress**
- bool **writeOnRelease**
- bool **writeOnClick**
- QString **pressText**
- QString **releaseText**
- QString **clickText**
- QString **clickCheckedText**
- QString **labelText**
- QString **program**
- QStringList **arguments**
- QString **guiFile**
- CreationOptionNames **creationOption**

9.82.1 Member Function Documentation

9.82.1.1 void QERadioButton::dbValueChanged (const QString & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.82.1.2 void QERadioButton::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.82.2 Property Documentation

9.82.2.1 bool QERadioButton::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.82.2.2 bool QERadioButton::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.82.2.3 unsigned QERadioButton::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.82.2.4 bool QERadioButton::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.82.2.5 UserLevels QERadioButton::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.82.2.6 QString QERadioButton::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.82.2.7 QString QERadioButton::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For

example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.82.2.8 QString QERadioButton::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.82.2.9 UserLevels QERadioButton::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.82.2.10 QString QERadioButton::variable [read, write]

EPICS variable name (CA PV)

9.82.2.11 bool QERadioButton::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.82.2.12 QString QERadioButton::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.82.2.13 bool QERadioButton::visible [read, write]

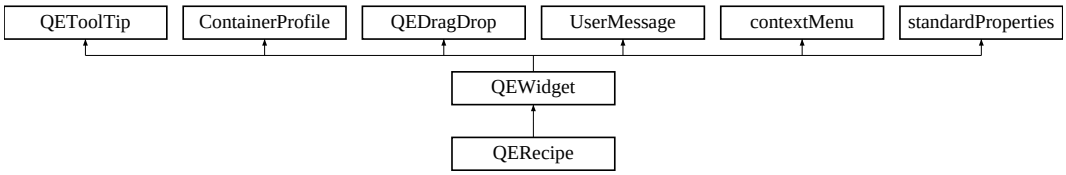
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEButton/QERadioButton.h
- /home/andrew/epicsqt/framework/widgets/QEButton/QERadioButton.cpp

9.83 QERecipe Class Reference

Inheritance diagram for QERecipe:



Public Types

- enum **configurationTypesProperty** { **File** = FROM_FILE, **Text** = FROM_TEXT }
- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **userTypesProperty** { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

Public Member Functions

- **QERecipe** (QWidget ∗pParent=0)
- void **setRecipeDescription** (QString pValue)
- QString **getRecipeDescription** ()
- void **setShowRecipeList** (bool pValue)
- bool **getShowRecipeList** ()
- void **setShowNew** (bool pValue)
- bool **getShowNew** ()
- void **setShowSave** (bool pValue)
- bool **getShowSave** ()
- void **setShowDelete** (bool pValue)
- bool **getShowDelete** ()
- void **setShowApply** (bool pValue)
- bool **getShowApply** ()
- void **setShowRead** (bool pValue)
- bool **getShowRead** ()
- void **setShowFields** (bool pValue)
- bool **getShowFields** ()

- void **setConfigurationType** (int pValue)
- int **getConfigurationType** ()
- void **setConfigurationFile** (QString pValue)
- QString **getConfigurationFile** ()
- void **setRecipeFile** (QString pValue)
- QString **getRecipeFile** ()
- void **setConfigurationText** (QString pValue)
- QString **getConfigurationText** ()
- void **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- void **setCurrentUserType** (int pValue)
- int **getCurrentUserType** ()
- bool **saveRecipeList** ()
- void **refreshRecipeList** ()
- void **refreshButton** ()
- void **userLevelChanged** (userLevels pValue)
- void **setConfigurationTypeProperty** (configurationTypesProperty pConfiguration-Type)
- configurationTypesProperty **getConfigurationTypeProperty** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()
- void **setCurrentUserTypeProperty** (userTypesProperty pUserType)
- userTypesProperty **getCurrentUserTypeProperty** ()

Protected Attributes

- QLabel * **qLabelRecipeDescription**
- QComboBox * **qComboBoxRecipeList**
- QPushButton * **qPushButtonNew**
- QPushButton * **qPushButtonSave**
- QPushButton * **qPushButtonDelete**
- QPushButton * **qPushButtonApply**
- QPushButton * **qPushButtonRead**
- QEConfiguredLayout * **qEConfiguredLayoutRecipeFields**
- QDomDocument **document**
- QString **recipeFile**
- QString **filename**
- int **detailsLayout**
- int **currentUserType**

Properties

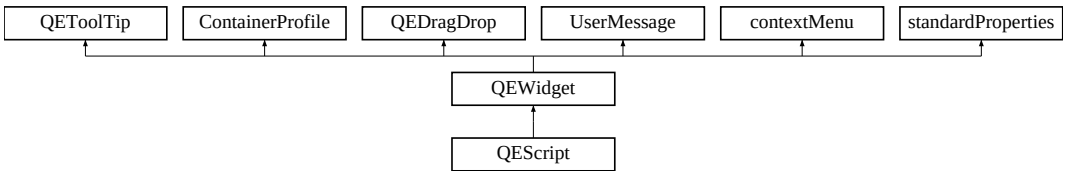
- QString **recipeDescription**
- bool **showRecipeList**
- bool **showNew**
- bool **showSave**
- bool **showDelete**
- bool **showApply**
- bool **showRead**
- bool **showFields**
- configurationTypesProperty **configurationType**
- QString **configurationFile**
- QString **configurationText**
- detailsLayoutProperty **detailsLayout**
- userTypesProperty **currentUserType**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QERecipe/QERecipe.h
- /home/andrew/epicsqt/framework/widgets/QERecipe/QERecipe.cpp

9.84 QEScript Class Reference

Inheritance diagram for QEScript:



Public Types

- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }

Signals

- void **selected** (QString pFilename)

Public Member Functions

- **QEScript** (QWidget *pParent=0)
- void **setDirectoryPath** (QString pValue)
- QString **getDirectoryPath** ()
- void **setShowDirectoryPath** (bool pValue)
- bool **getShowDirectoryPath** ()
- void **setShowDirectoryBrowser** (bool pValue)
- bool **getShowDirectoryBrowser** ()
- void **setShowRefresh** (bool pValue)
- bool **getShowRefresh** ()
- void **setShowColumnTime** (bool pValue)
- bool **getShowColumnTime** ()
- void **setShowColumnSize** (bool pValue)
- bool **getShowColumnSize** ()
- void **setShowColumnFilename** (bool pValue)
- bool **getShowColumnFilename** ()
- void **setShowFileExtension** (bool pValue)
- bool **getShowFileExtension** ()
- void **setFileFilter** (QString pValue)
- QString **getFileFilter** ()
- void **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- void **updateTable** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()

Protected Attributes

- QLineEdit * **qlineEditDirectoryPath**
- QPushButton * **qPushButtonDirectoryBrowser**
- QPushButton * **qPushButtonRefresh**
- _QTableWidgetScript * **qTableWidgetScript**
- QString **fileFilter**
- bool **showFileExtension**
- int **detailsLayout**

Properties

- QString **directoryPath**
- bool **showDirectoryPath**
- bool **showDirectoryBrowser**
- bool **showRefresh**
- bool **showColumnTime**
- bool **showColumnSize**
- bool **showColumnFilename**

- detailsLayoutProperty **detailsLayout**

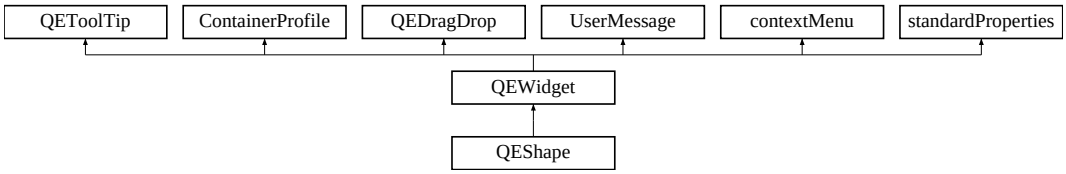
The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEScript/QEScript.h
- /home/andrew/epicsqt/framework/widgets/QEScript/QEScript.cpp

9.85 QEShape Class Reference

`#include <QEShape.h>`

Inheritance diagram for QEShape:



Public Types

- enum shapeOptions { **Line**, **Points**, **Polyline**, **Polygon**, **Rect**, **RoundedRect**, **Ellipse**, **Arc**, **Chord**, **Pie**, **Path**, **Text**, **Pixmap** }
- enum animationOptions { **Width**, **Height**, **X**, **Y**, **Transperency**, **Rotation**, **ColourHue**, **ColourSaturation**, **ColourValue**, **ColourIndex**, **Penwidth** }
- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Public Slots

- void requestEnabled (const bool &state)

Signals

- void dbValueChanged1 (const qlonglong &out)
- void dbValueChanged2 (const qlonglong &out)
- void dbValueChanged3 (const qlonglong &out)
- void dbValueChanged4 (const qlonglong &out)
- void dbValueChanged5 (const qlonglong &out)
- void dbValueChanged6 (const qlonglong &out)

Public Member Functions

- QEShape (QWidget *parent=0)
- QEShape (const QString &variableName, QWidget *parent=0)
- void setAnimation (animationOptions animation, const int index)

  Access function for 'animation' properties - refer to 'animation' properties for details.
- animationOptions getAnimation (const int index)

  Access function for 'animation' properties - refer to 'animation' properties for details.
- void setScale (const double scale, const int index)

  Access function for 'scale' properties - refer to 'scale' properties for details.
- double getScale (const int index)

  Access function for 'scale' properties - refer to 'scale' properties for details.
- void setOffset (const double offset, const int index)

  Access function for 'offset' properties - refer to 'offset' properties for details.
- double getOffset (const int index)

  Access function for 'offset' properties - refer to 'offset' properties for details.
- void setBorder (const bool border)

  Access function for 'border' properties - refer to 'border' properties for details.
- bool getBorder ()

  Access function for 'border' properties - refer to 'border' properties for details.
- void setFill (const bool fill)

  Access function for 'fill' properties - refer to 'fill' properties for details.
- bool getFill ()

  Access function for 'fill' properties - refer to 'fill' properties for details.
- void setShape (shapeOptions shape)

  Access function for 'shape' properties - refer to 'shape' properties for details.
- shapeOptions getShape ()

  Access function for 'shape' properties - refer to 'shape' properties for details.
- void setNumPoints (const unsigned int numPoints)

  Access function for 'number of points' properties - refer to 'number of points' properties for details.
- unsigned int getNumPoints ()

  Access function for 'number of points' properties - refer to 'number of points' properties for details.
- void setOriginTranslation (const QPoint originTranslation)

  Access function for 'origin translation' properties - refer to 'origin translation' properties for details.
- QPoint getOriginTranslation ()

  Access function for 'origin translation' properties - refer to 'origin translation' properties for details.
- void setPoint (const QPoint point, const int index)

  Access function for 'point' properties - refer to 'point' properties for details.
- QPoint getPoint (const int index)

  Access function for 'point' properties - refer to 'point' properties for details.

- void setColor (const QColor color, const int index)

  Access function for 'colour' properties - refer to 'colour' properties for details.
- QColor getColor (const int index)

  Access function for 'colour' properties - refer to 'colour' properties for details.
- void setDrawBorder (const bool drawBorder)

  Access function for 'draw border' properties - refer to 'draw border' properties for details.
- bool getDrawBorder ()

  Access function for 'draw border' properties - refer to 'draw border' properties for details.
- void setLineWidth (const unsigned int lineWidth)

  Access function for 'line width' properties - refer to 'line width' properties for details.
- unsigned int getLineWidth ()

  Access function for 'line width' properties - refer to 'line width' properties for details.
- void setStartAngle (const double startAngle)

  Access function for 'start angle' properties - refer to 'start angle' properties for details.
- double getStartAngle ()

  Access function for 'start angle' properties - refer to 'start angle' properties for details.
- void setRotation (const double rotation)

  Access function for 'rotation' properties - refer to 'rotation' properties for details.
- double getRotation ()

  Access function for 'rotation' properties - refer to 'rotation' properties for details.
- void setArcLength (const double arcLength)

  Access function for 'arc length' properties - refer to 'arc length' properties for details.
- double getArcLength ()

  Access function for 'arc length' properties - refer to 'arc length' properties for details.
- void setText (const QString text)

  Access function for 'text' properties - refer to 'text' properties for details.
- QString getText ()

  Access function for 'text' properties - refer to 'text' properties for details.
- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.

Properties

- QString variable1
- QString variable2
- QString variable3
- QString variable4
- QString variable5
- QString variable6
- QString variableSubstitutions
- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- animationOptions animation1
- animationOptions animation2
- animationOptions animation3
- animationOptions animation4
- animationOptions animation5
- animationOptions animation6
- double scale1

  Scale factor applied to data from the 1st variable before it is used to animate the shape.
- double scale2
- double scale3
- double scale4
- double scale5
- double scale6
- double offset1
- double offset2
- double offset3
- double offset4
- double offset5
- double offset6
- QPoint point1
- QPoint point2
- QPoint point3
- QPoint point4
- QPoint point5
- QPoint point6
- QPoint point7
- QPoint point8

- QPoint point9
- QPoint point10
- QColor color1
- QColor color2
- QColor color3
- QColor color4
- QColor color5
- QColor color6
- QColor color7
- QColor color8
- QColor color9
- QColor color10

9.85.1 Detailed Description

This class is a EPICS aware shape widget based on the Qt widget. One of several shapes can be drawn within the widget, and up to 6 variables can be used to animate various attributes of the shape. For example to represent beam positino and size, an elipse can be drawn with four variables animating its vertcal and horizontal size and position. It is tighly integrated with the base class QEWidget which provides generic support such as macro substitutions, drag/drop, and standard properties.

9.85.2 Member Enumeration Documentation

9.85.2.1 enum QEShape::animationOptions

Options for how a variable will animate the shape.

9.85.2.2 enum QEShape::shapeOptions

Options for the type of shape.

9.85.3 Constructor & Destructor Documentation

9.85.3.1 QEShape::QEShape (QWidget * *parent* = 0)

Create without a variable. Use setVariableNameProperty() and setSubstitutions-Property() to define a variable and, optionally, macro substitutions later.

9.85.3.2 QEShape::QEShape (const QString & *variableName,* QWidget * *parent* = 0)

Create with a single variable. (Note, the QEShape widget can use up to 6 variables) A connection is automatically established. If macro substitutions are required, create without a variable and set the variable and macro substitutions after creation.

9.85.4 Member Function Documentation

9.85.4.1 void QEShape::dbValueChanged1 (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change for the first variable Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.85.4.2 void QEShape::dbValueChanged2 (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change for the second variable Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.85.4.3 void QEShape::dbValueChanged3 (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change for the third variable Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.85.4.4 void QEShape::dbValueChanged4 (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change for the fourth variable Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.85.4.5 void QEShape::dbValueChanged5 (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change for the fifth variable Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.85.4.6 void QEShape::dbValueChanged6 (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change for the sixth variable Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.85.4.7 void QEShape::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.85.5 Property Documentation

9.85.5.1 bool QEShape::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.85.5.2 animationOptions QEShape::animation1 [read, write]

Animation to be effected by the 1st variable. This is used to select what the effect changing data for the 1st variable will have on the shape.

9.85.5.3 animationOptions QEShape::animation2 [read, write]

Animation to be effected by the 2nd variable. This is used to select what the effect changing data for the 2nd variable will have on the shape.

9.85.5.4 animationOptions QEShape::animation3 [read, write]

Animation to be effected by the 3rd variable. This is used to select what the effect changing data for the 3rd variable will have on the shape.

9.85.5.5 animationOptions QEShape::animation4 [read, write]

Animation to be effected by the 4th variable. This is used to select what the effect changing data for the 4th variable will have on the shape.

9.85.5.6 animationOptions QEShape::animation5 [read, write]

Animation to be effected by the 5th variable. This is used to select what the effect changing data for the 5th variable will have on the shape.

9.85.5.7 animationOptions QEShape::animation6 [read, write]

Animation to be effected by the 6th variable. This is used to select what the effect changing data for the 6th variable will have on the shape.

9.85.5.8 QColor QEShape::color1 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.85.5.9 QColor QEShape::color10 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.85.5.10 QColor QEShape::color2 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.85.5.11 QColor QEShape::color3 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.85.5.12 QColor QEShape::color4 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.85.5.13 QColor QEShape::color5 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.85.5.14 QColor QEShape::color6 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.85.5.15 QColor QEShape::color7 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.85.5.16 QColor QEShape::color8 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.85.5.17 QColor QEShape::color9 [read, write]

Used by the color animation to determine the color based on a data value. The scaled and offset data is used as an index to select color properties 'color1' to 'color10'.

9.85.5.18 bool QEShape::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.85.5.19 unsigned QEShape::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

The number of points to use when drawing shapes that are defined by a variable number of points, such as polyline, polygon, path, and series of points.

Sets the width of the pen. Used for the following shapes: Line, Points, Polyline, Polygon, Rect, RoundedRect, Ellipse, Arc, Chord, Pie, Path

9.85.5.20 double QEShape::offset1 [read, write]

Offset applied to data from the 1st variable before it is used to animate the shape

9.85.5.21 double QEShape::offset2 [read, write]

Offset applied to data from the 2nd variable before it is used to animate the shape

9.85.5.22 double QEShape::offset3 [read, write]

Offset applied to data from the 3rd variable before it is used to animate the shape

9.85.5.23 double QEShape::offset4 [read, write]

Offset applied to data from the 4th variable before it is used to animate the shape

9.85.5.24 double QEShape::offset5 [read, write]

Offset applied to data from the 5th variable before it is used to animate the shape

9.85.5.25 double QEShape::offset6 [read, write]

Offset applied to data from the 6th variable before it is used to animate the shape

9.85.5.26 QPoint QEShape::point1 [read, write]

1st coordinate used when drawing the shape. Used for the following shapes: Line, Points, Polyline, Polygon, Rect, RoundedRect, Ellipse, Arc, Chord, Pie, Path, Text, - Pixmap

9.85.5.27 QPoint QEShape::point10 [read, write]

10th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.85.5.28 QPoint QEShape::point2 [read, write]

2nd coordinate used when drawing the shape. Used for the following shapes: Line, Points, Polyline, Polygon, Rect, RoundedRect, Ellipse, Arc, Chord, Pie, Path, Pixmap

9.85.5.29 QPoint QEShape::point3 [read, write]

3rd coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.85.5.30 QPoint QEShape::point4 [read, write]

4th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.85.5.31 QPoint QEShape::point5 [read, write]

5th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.85.5.32 QPoint QEShape::point6 [read, write]

6th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.85.5.33 QPoint QEShape::point7 [read, write]

7th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.85.5.34 QPoint QEShape::point8 [read, write]

8th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.85.5.35 QPoint QEShape::point9 [read, write]

9th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.85.5.36 double QEShape::scale2 [read, write]

Scale factor applied to data from the 2nd variable before it is used to animate the shape

9.85.5.37 double QEShape::scale3 [read, write]

Scale factor applied to data from the 3rd variable before it is used to animate the shape

9.85.5.38 double QEShape::scale4 [read, write]

Scale factor applied to data from the 4th variable before it is used to animate the shape

9.85.5.39 double QEShape::scale5 [read, write]

Scale factor applied to data from the 5th variable before it is used to animate the shape

9.85.5.40 double QEShape::scale6 [read, write]

Scale factor applied to data from the 6th variable before it is used to animate the shape

9.85.5.41 UserLevels QEShape::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through set-UserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.85.5.42 QString QEShape::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the style-Manager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.85.5.43 QString QEShape::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the style-Manager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.85.5.44 QString QEShape::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.85.5.45 UserLevels QEShape::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.85.5.46 QString QEShape::variable1 [read, write]

EPICS variable name (CA PV). This variable is read and used to animate an attribute of the shape. The value read is first scaled and offset by properties scale1 and offset1 then the attribute selected for animation is selected by the property animation1.

9.85.5.47 QString QEShape::variable2 [read, write]

EPICS variable name (CA PV). This variable is read and used to animate an attribute of the shape. The value read is first scaled and offset by properties scale2 and offset2

then the attribute selected for animation is selected by the property animation2.

9.85.5.48 QString QEShape::variable3 [read, write]

EPICS variable name (CA PV). This variable is read and used to animate an attribute of the shape. The value read is first scaled and offset by properties scale3 and offset3 then the attribute selected for animation is selected by the property animation3.

9.85.5.49 QString QEShape::variable4 [read, write]

EPICS variable name (CA PV). This variable is read and used to animate an attribute of the shape. The value read is first scaled and offset by properties scale4 and offset4 then the attribute selected for animation is selected by the property animation4.

9.85.5.50 QString QEShape::variable5 [read, write]

EPICS variable name (CA PV). This variable is read and used to animate an attribute of the shape. The value read is first scaled and offset by properties scale5 and offset5 then the attribute selected for animation is selected by the property animation5.

9.85.5.51 QString QEShape::variable6 [read, write]

EPICS variable name (CA PV). This variable is read and used to animate an attribute of the shape. The value read is first scaled and offset by properties scale6 and offset6 then the attribute selected for animation is selected by the property animation6.

9.85.5.52 bool QEShape::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.85.5.53 QString QEShape::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'SAMPLE=SAM1, NAME = "Ref foil"' These substitutions are applied to all the variable names.

9.85.5.54 bool QEShape::visible [read, write]

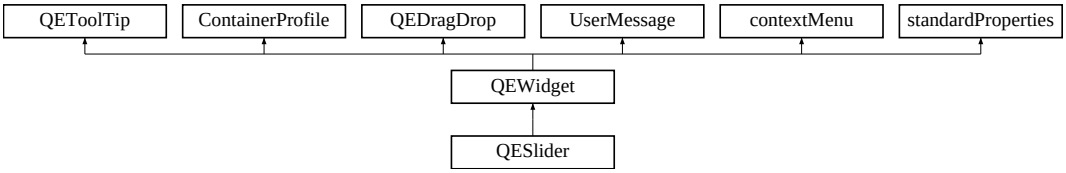
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEShape/QEShape.h
- /home/andrew/epicsqt/framework/widgets/QEShape/QEShape.cpp

9.86 QESlider Class Reference

Inheritance diagram for QESlider:



Public Types

- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Public Slots

- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const qlonglong &out)

Public Member Functions

- **QESlider** (QWidget ∗parent=0)
- **QESlider** (const QString &variableName, QWidget ∗parent=0)
- void **setWriteOnChange** (bool writeOnChange)
- bool **getWriteOnChange** ()
- void **setSubscribe** (bool subscribe)
- bool **getSubscribe** ()
- void **setScale** (double scaleIn)
- double **getScale** ()
- void **setOffset** (double offsetIn)

- double **getOffset** ()
- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.
- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.
- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Protected Attributes

- QEFloatingFormatting **floatingFormatting**
- bool writeOnChange

Properties

- QString variable
- QString variableSubstitutions
- bool subscribe
- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled

9.86.1 Member Function Documentation

9.86.1.1 void QESlider::dbValueChanged (const qlonglong & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.86.1.2 void QESlider::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.86.2 Member Data Documentation

9.86.2.1 bool QESlider::writeOnChange [read, write, protected]

Sets if this widget writes any changes as the user moves the slider (the QSlider 'valueChanged' signal is emitted). Default is 'true' (writes any changes when the QSlider 'valueChanged' signal is emitted).

9.86.3 Property Documentation

9.86.3.1 bool QESlider::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.86.3.2 bool QESlider::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.86.3.3 unsigned QESlider::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.86.3.4 bool QESlider::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.86.3.5 UserLevels QESlider::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.86.3.6 QString QESlider::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.86.3.7 QString QESlider::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.86.3.8 QString QESlider::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.86.3.9 UserLevels QESlider::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.86.3.10 QString QESlider::variable [read, write]

EPICS variable name (CA PV)

9.86.3.11 bool QESlider::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.86.3.12 QString QESlider::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.86.3.13 bool QESlider::visible [read, write]

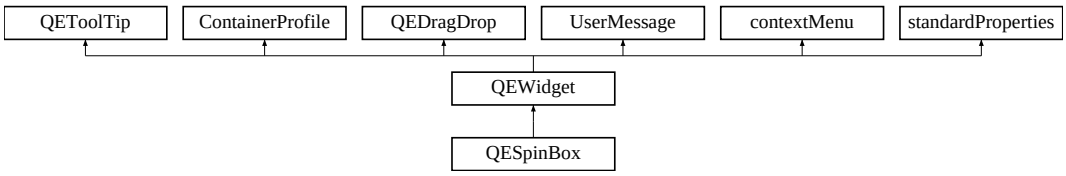
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QESlider/QESlider.h
- /home/andrew/epicsqt/framework/widgets/QESlider/QESlider.cpp

9.87 QESpinBox Class Reference

Inheritance diagram for QESpinBox:



Public Types

- enum UserLevels { **User** = USERLEVEL_USER, **Scientist** = USERLEVEL_SCIENTIST, **Engineer** = USERLEVEL_ENGINEER }

  User friendly enumerations for userLevelVisibility and userLevelEnabled properties - refer to userLevelVisibility and userLevelEnabled properties and userLevel enumeration for details.

Public Slots

- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const double &out)
- void userChange (const QString &oldValue, const QString &newValue, const QString &lastValue)

  Internal use only. Used by QEConfiguredLayout to be notified when one of its widgets has written something.

Public Member Functions

- **QESpinBox** (QWidget ∗parent=0)
- **QESpinBox** (const QString &variableName, QWidget ∗parent=0)
- void **setWriteOnChange** (bool writeOnChangeIn)
- bool **getWriteOnChange** ()
- void **setSubscribe** (bool subscribe)
- bool **getSubscribe** ()
- void **setAddUnitsAsSuffix** (bool addUnitsAsSuffixIn)
- bool **getAddUnitsAsSuffix** ()
- void **setUseDbPrecisionForDecimals** (bool useDbPrecisionForDecimalIn)
- bool **getUseDbPrecisionForDecimals** ()
- bool isEnabled () const

  Access function for 'enabled' property - refer to 'enabled' property for details.
- void setEnabled (bool state)

  Access function for 'enabled' property - refer to 'enabled' property for details.
- UserLevels getUserLevelVisibilityProperty ()

Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.

- void setUserLevelVisibilityProperty (UserLevels level)

  Access function for 'userLevelVisibility' property - refer to 'userLevelVisibility' property for details.

- UserLevels getUserLevelEnabledProperty ()

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.

- void setUserLevelEnabledProperty (UserLevels level)

  Access function for 'userLevelEnabled' property - refer to 'userLevelEnabled' property for details.

Protected Member Functions

- void **establishConnection** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent ∗event)
- void **dropEvent** (QDropEvent ∗event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()

Protected Attributes

- QEFloatingFormatting **floatingFormatting**
- bool **writeOnChange**
- bool **addUnitsAsSuffix**
- bool **useDbPrecisionForDecimal**

Properties

- QString variable
- QString variableSubstitutions
- bool variableAsToolTip
- bool enabled
- bool allowDrop
- bool visible
- unsigned int
- QString userLevelUserStyle
- QString userLevelScientistStyle
- QString userLevelEngineerStyle
- UserLevels userLevelVisibility
- UserLevels userLevelEnabled
- bool subscribe
- bool **useDbPrecision**
- bool **addUnits**

9.87.1 Member Function Documentation

9.87.1.1 void QESpinBox::dbValueChanged (const double & *out*) [signal]

Sent when the widget is updated following a data change Can be used to pass on EPICS data (as presented in this widget) to other widgets. For example a QList widget could log updates from this widget.

9.87.1.2 void QESpinBox::requestEnabled (const bool & *state*) [inline, slot]

Similar to standard setEnabled slot, but allows QE widget to determine if the widget remains disabled due to invalid data. If disabled due to invalid data, a request to enable the widget will be honoured when the data is no longer invalid.

9.87.2 Property Documentation

9.87.2.1 bool QESpinBox::allowDrop [read, write]

Allow drag/drops operations to this widget. Default is false. Any dropped text will be used as a new variable name.

Reimplemented from QEDragDrop.

9.87.2.2 bool QESpinBox::enabled [read, write]

Set the prefered 'enabled' state. Default is true. This property is copied to the standard Qt 'enabled' property if the data being displayed is valid. If the data being displayed is invalid the standard Qt 'enabled' property will always be set to false to indicate invalid data. The value of this property will only be copied to the standard Qt 'enabled' property once data is valid.

9.87.2.3 unsigned QESpinBox::int [read, write]

Set the ID used by the message filtering system. Default is zero. Widgets or applications that use messages from the framework have the option of filtering on this ID. For example, by using a unique message source ID a QELog widget may be set up to only log messages from a select set of widgets.

9.87.2.4 bool QESpinBox::subscribe [read, write]

Sets if this widget subscribes for data updates and displays current data. Default is 'true' (subscribes for and displays data updates)

Reimplemented from QEWidget.

9.87.2.5 UserLevels QESpinBox::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always accessable should be visible at 'User'. Widgets that are only accessable to scientists managing the facility should be visible at 'Scientist'. - Widgets that are only accessable to engineers maintaining the facility should be visible at 'Engineer'.

9.87.2.6 QString QESpinBox::userLevelEngineerStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Engineer' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.87.2.7 QString QESpinBox::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. - Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.87.2.8 QString QESpinBox::userLevelUserStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'User' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the styleManager class. Refer to the styleManager class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.87.2.9 UserLevels QESpinBox::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application through the QELogin widget, or programatically through setUserLevel() Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.87.2.10 QString QESpinBox::variable [read, write]

EPICS variable name (CA PV)

9.87.2.11 bool QESpinBox::variableAsToolTip [read, write]

Use the variable as the tool tip. Default is true. Tool tip property will be overwritten by the variable name.

Reimplemented from QEWidget.

9.87.2.12 QString QESpinBox::variableSubstitutions [read, write]

Macro substitutions. The default is no substitutions. The format is NAME1=VALUE1[,] NAME2=VALUE2... Values may be quoted strings. For example, 'PUMP=PMP3, NAME = "My Pump"' These substitutions are applied to variable names for all QE widgets. In some widgets are are also used for other purposes.

9.87.2.13 bool QESpinBox::visible [read, write]

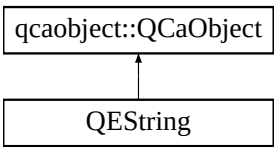
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QESpinBox/QESpinBox.h
- /home/andrew/epicsqt/framework/widgets/QESpinBox/QESpinBox.cpp

9.88 QEString Class Reference

Inheritance diagram for QEString:



Public Slots

- void **writeString** (const QString &data)

Signals

- void **stringConnectionChanged** (QCaConnectionInfo &connectionInfo, const unsigned int &variableIndex)
- void **stringChanged** (const QString &value, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)

Public Member Functions

- **QEString** (QString recordName, QObject *eventObject, QEStringFormatting *stringFormattingIn, unsigned int variableIndexIn)
- **QEString** (QString recordName, QObject *eventObject, QEStringFormatting *stringFormattingIn, unsigned int variableIndexIn, UserMessage *userMessageIn)
- bool **writeString** (const QString &data, QString &message)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QEString.h
- /home/andrew/epicsqt/framework/data/src/QEString.cpp

9.89 QEStringFormatting Class Reference

Public Types

- enum **formats** { **FORMAT_DEFAULT**, **FORMAT_FLOATING**, **FORMAT_INTEGER**, **FORMAT_UNSIGNEDINTEGER**, **FORMAT_TIME**, **FORMAT_LOCAL_ENUMERATE**, **FORMAT_STRING** }
- enum **notations** { **NOTATION_FIXED** = QTextStream::FixedNotation, **NOTATION_SCIENTIFIC** = QTextStream::ScientificNotation, **NOTATION_AUTOMATIC** = QTextStream::SmartNotation }
- enum **arrayActions** { **APPEND**, **ASCII**, **INDEX** }

Public Member Functions

- QString **formatString** (const QVariant &value)
- QVariant **formatValue** (const QString &text, bool &ok)
- void **setDbEgu** (QString egu)
- void **setDbEnumerations** (QStringList enumerations)
- void **setDbPrecision** (unsigned int dbPrecisionIn)
- void **setDbVariableIsStatField** (bool isStatField)
- void **setPrecision** (int precision)
- void **setUseDbPrecision** (bool useDbPrecision)
- void **setLeadingZero** (bool leadingZero)
- void **setTrailingZeros** (bool trailingZeros)

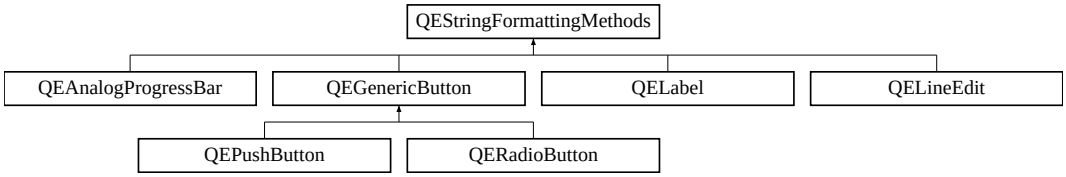
- void **setFormat** (formats format)
- void **setRadix** (unsigned int radix)
- void **setNotation** (notations notation)
- void **setArrayAction** (arrayActions arrayActionIn)
- void **setArrayIndex** (unsigned int arrayIndexIn)
- void **setAddUnits** (bool addUnits)
- void **setLocalEnumeration** (QString localEnumerationIn)
- int **getPrecision** ()
- bool **getUseDbPrecision** ()
- bool **getLeadingZero** ()
- bool **getTrailingZeros** ()
- formats **getFormat** ()
- unsigned int **getRadix** ()
- notations **getNotation** ()
- arrayActions **getArrayAction** ()
- unsigned int **getArrayIndex** ()
- bool **getAddUnits** ()
- QString **getLocalEnumeration** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QEStringFormatting.h
- /home/andrew/epicsqt/framework/data/src/QEStringFormatting.cpp

9.90 QEStringFormattingMethods Class Reference

Inheritance diagram for QEStringFormattingMethods:



Public Member Functions

- virtual void **stringFormattingChange** ()=0
- void **setPrecision** (int precision)
- int **getPrecision** ()
- void **setUseDbPrecision** (bool useDbPrecision)
- bool **getUseDbPrecision** ()
- void **setLeadingZero** (bool leadingZero)
- bool **getLeadingZero** ()

- void **setTrailingZeros** (bool trailingZeros)
- bool **getTrailingZeros** ()
- void **setAddUnits** (bool addUnits)
- bool **getAddUnits** ()
- void **setLocalEnumeration** (QString localEnumeration)
- QString **getLocalEnumeration** ()
- void **setFormat** (QEStringFormatting::formats format)
- QEStringFormatting::formats **getFormat** ()
- void **setRadix** (unsigned int radix)
- unsigned int **getRadix** ()
- void **setNotation** (QEStringFormatting::notations notation)
- QEStringFormatting::notations **getNotation** ()
- void **setArrayAction** (QEStringFormatting::arrayActions arrayAction)
- QEStringFormatting::arrayActions **getArrayAction** ()
- void **setArrayIndex** (unsigned int arrayIndex)
- unsigned int **getArrayIndex** ()

Protected Attributes

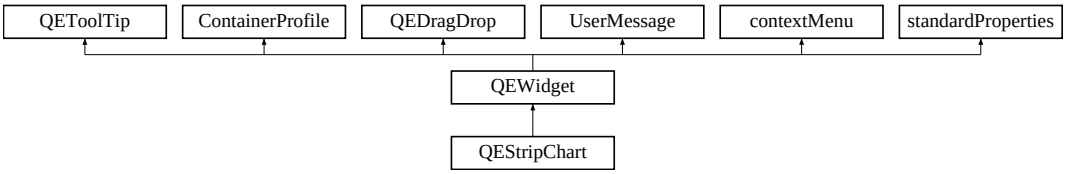
- QEStringFormatting **stringFormatting**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/QEStringFormattingMethods.-h
- /home/andrew/epicsqt/framework/widgets/src/QEStringFormattingMethods.cpp

9.91 QEStripChart Class Reference

Inheritance diagram for QEStripChart:



Classes

- class PrivateData

Public Types

- enum **Constants** { **NUMBER_OF_PVS** = 12 }

Public Member Functions

- **QEStripChart** (QWidget *parent=0)
- QSize **sizeHint** () const
- QDateTime **getStartDateTime** ()
- QDateTime **getEndDateTime** ()
- void **setEndDateTime** (QDateTime endDateTimeIn)
- int **getDuration** ()
- void **setDuration** (int durationIn)
- double **getYMinimum** ()
- void **setYMinimum** (double yMinimumIn)
- double **getYMaximum** ()
- void **setYMaximum** (double yMaximumIn)
- void **plotData** ()

Protected Member Functions

- void **dragEnterEvent** (QDragEnterEvent *event)
- void **dropEvent** (QDropEvent *event)
- void **mousePressEvent** (QMouseEvent *event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)
- void **setup** ()
- qcaobject::QCaObject * **createQcaItem** (unsigned int variableIndex)
- void **establishConnection** (unsigned int variableIndex)
- void **updateToolTip** (const QString &tip)

Properties

- int **duration**
- double **yMinimum**
- double **yMaximum**
- QString **variable1**
- QString **variable2**
- QString **variable3**
- QString **variable4**
- QString **variable5**
- QString **variable6**
- QString **variable7**
- QString **variable8**
- QString **variable9**
- QString **variable10**
- QString **variable11**

- QString **variable12**
- QColor **colour1**
- QColor **colour2**
- QColor **colour3**
- QColor **colour4**
- QColor **colour5**
- QColor **colour6**
- QColor **colour7**
- QColor **colour8**
- QColor **colour9**
- QColor **colour10**
- QColor **colour11**
- QColor **colour12**

Friends

- class **PrivateData**
- class **QEStripChartItem**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChart.h
- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChart.cpp

9.92 QEStripChartItem Class Reference

Classes

- class PrivateData

Friends

- class **QEStripChart**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChartItem.h
- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChartItem.cpp

9.93 QEStripChartItemDialog Class Reference

Public Member Functions

- **QEStripChartItemDialog** (QWidget ∗parent=0)
- void **setPvName** (QString pvNameIn)
- QString **getPvName** ()
- void **setColour** (QColor colourIn)
- QColor **getColour** ()
- bool **isClear** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChartItem-Dialog.h
- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChartItem-Dialog.cpp

9.94 QEStripChartTimeDialog Class Reference

Public Member Functions

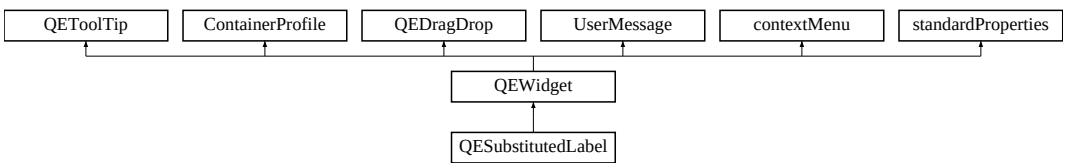
- **QEStripChartTimeDialog** (QWidget ∗parent=0)
- void **setMaximumDateTime** (QDateTime datetime)
- void **setStartDateTime** (QDateTime datetime)
- QDateTime **getStartDateTime** ()
- void **setEndDateTime** (QDateTime datetime)
- QDateTime **getEndDateTime** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChartTime-Dialog.h
- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChartTime-Dialog.cpp

9.95 QESubstitutedLabel Class Reference

Inheritance diagram for QESubstitutedLabel:



Public Member Functions

- **QESubstitutedLabel** (QWidget *parent=0)
- void **establishConnection** (unsigned int variableIndex)
- void **setLabelTextProperty** (QString labelTextIn)
- QString **getLabelTextProperty** ()
- QString **getLabelTextPropertyFormat** ()
- void **setLabelTextPropertyFormat** (QString labelTextIn)

Protected Attributes

- QString labelText

Properties

- QString textSubstitutions

9.95.1 Member Data Documentation

9.95.1.1 QString QESubstitutedLabel::labelText [read, write, protected]

Label text to be substituted. This text will be copied to the label text after applying any macro substitutions from the textSubstitutions property

9.95.2 Property Documentation

9.95.2.1 QString QESubstitutedLabel::textSubstitutions [read, write]

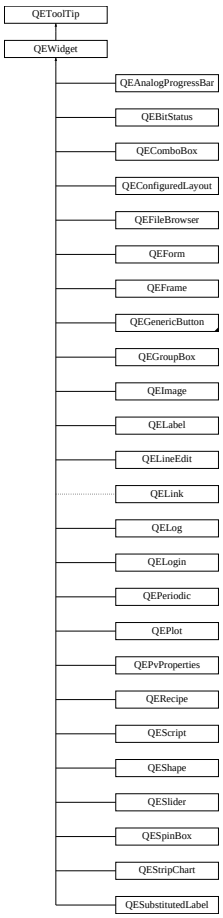
Text substitutions. These substitutions are applied to the 'labelText' property prior to copying it to the label text.

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QESubstitutedLabel/QESubstituted-Label.h
- /home/andrew/epicsqt/framework/widgets/QESubstitutedLabel/QESubstituted-Label.cpp

9.96 QEToolTip Class Reference

Inheritance diagram for QEToolTip:



Public Member Functions

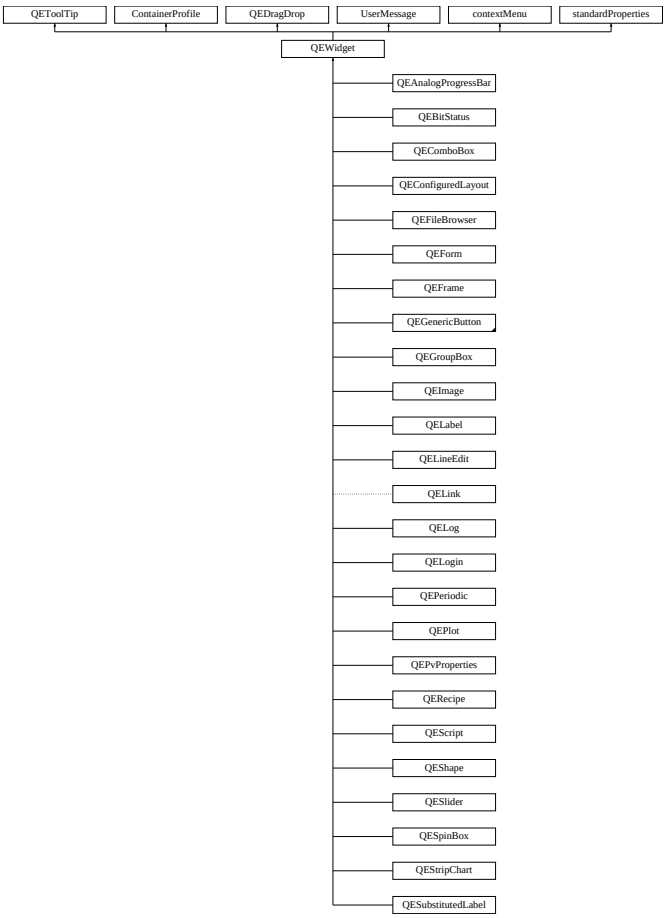
- void **updateToolTipVariable** (const QString &variable)
- void **updateToolTipAlarm** (const QString &alarm)
- void **updateToolTipConnection** (bool connection)
- virtual void **updateToolTip** (const QString &)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/QEToolTip.h
- /home/andrew/epicsqt/framework/widgets/src/QEToolTip.cpp

9.97 QEWidget Class Reference

Inheritance diagram for QEWidget:



Public Member Functions

- **QEWidget** (QWidget ∗ownerIn)
- void **activate** ()
- unsigned int **getMessageSourceId** ()
- void **setMessageSourceId** (unsigned int messageSourceId)
- qcaobject::QCaObject ∗ **getQcaItem** (unsigned int variableIndex)
- void **setupContextMenu** (QWidget ∗w)
- QColor **getColor** (QCaAlarmInfo &alarmInfo, const int saturation)
- void **readNow** ()
- virtual void **writeNow** ()
- virtual void **setVariableNameAndSubstitutions** (QString variableNameIn, QString variableNameSubstitutionsIn, unsigned int variableIndex)
- QFile ∗ **openQEFile** (QString name, QFile::OpenModeFlag mode)
- QString **defaultFileLocation** ()

Static Public Member Functions

- static bool **inDesigner** ()

Protected Member Functions

- void **setNumVariables** (unsigned int numVariablesIn)
- qcaobject::QCaObject * **createConnection** (unsigned int variableIndex)
- virtual qcaobject::QCaObject * **createQcaItem** (unsigned int variableIndex)
- virtual void **establishConnection** (unsigned int variableIndex)
- void **setVariableAsToolTip** (bool variableAsToolTip)
- bool **getVariableAsToolTip** ()

Protected Attributes

- bool **subscribe**
- bool **variableAsToolTip**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/QEWidget.h
- /home/andrew/epicsqt/framework/widgets/src/QEWidget.cpp

9.98 QEWidgets Class Reference

Public Member Functions

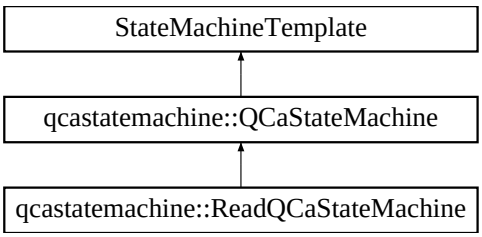
- **QEWidgets** (QObject *parent=0)
- virtual QList < QDesignerCustomWidgetInterface * > **customWidgets** () const

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/QEDesignerPlugin.h
- /home/andrew/epicsqt/framework/widgets/src/QEDesignerPlugin.cpp

9.99 qcastatemachine::ReadQCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::ReadQCaStateMachine:



Public Member Functions

- **ReadQCaStateMachine** (void ∗parent)
- bool **process** (int requestedState)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QCaStateMachine.h
- /home/andrew/epicsqt/framework/data/src/QCaStateMachine.cpp

9.100 RecordSpec Class Reference

Public Member Functions

- **RecordSpec** (const QString theRecordType)
- QString **getRecordType** ()
- QString **getFieldName** (const int index)

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEPvProperties/QEPvProperties.cpp

9.101 RecordSpecList Class Reference

Public Member Functions

- RecordSpec ∗ **find** (const QString recordType)

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEPvProperties/QEPvProperties.cpp

9.102 selectMenu Class Reference

Public Member Functions

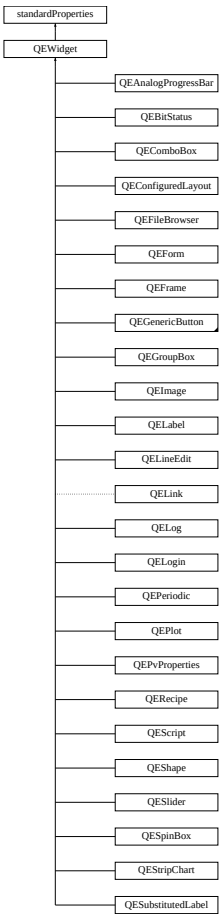
- **selectMenu** (QWidget ∗parent=0)
- imageContextMenu::imageContextMenuOptions **getSelectOption** (const QPoint &pos)
- void **setChecked** (const int mode)
- void **setPanEnabled** (bool enablePan)
- void **setVSliceEnabled** (bool enableVSliceSelection)
- void **setHSlicetEnabled** (bool enableHSliceSelection)
- void **setAreaEnabled** (bool enableAreaSelection)
- void **setProfileEnabled** (bool enableProfileSelection)
- void **setTargetEnabled** (bool enableTargetSelection)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/selectMenu.h
- /home/andrew/epicsqt/framework/widgets/QEImage/selectMenu.cpp

9.103 standardProperties Class Reference

Inheritance diagram for standardProperties:



Public Member Functions

- **standardProperties** (QWidget ∗ownerIn)

Protected Member Functions

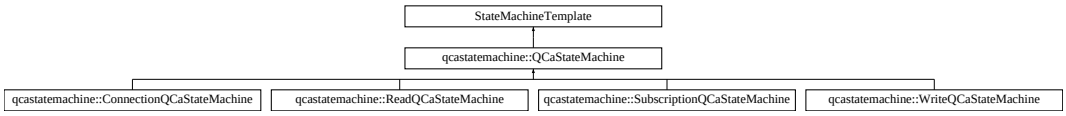
- userLevels **getUserLevelVisibility** ()
- void **setUserLevelVisibility** (userLevels level)
- userLevels **getUserLevelEnabled** ()
- void **setUserLevelEnabled** (userLevels level)
- bool **getApplicationEnabled** () const
- void **setApplicationEnabled** (bool state)
- void **setDataDisabled** (bool disable)
- void **setRunVisible** (bool visibleIn)
- bool **getRunVisible** ()
- void **checkVisibilityEnabledLevel** (userLevels level)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/standardProperties.h
- /home/andrew/epicsqt/framework/widgets/src/standardProperties.cpp

9.104 StateMachineTemplate Class Reference

Inheritance diagram for StateMachineTemplate:



Public Member Functions

- virtual bool **process** (int requestedState)=0

Public Attributes

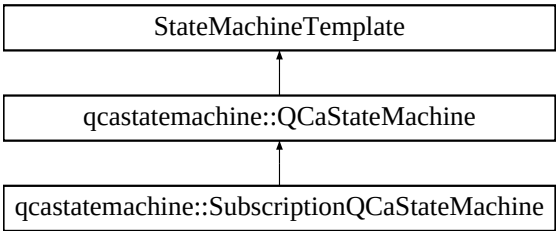
- int **currentState**
- int **requestState**

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/data/include/QCaStateMachine.h

9.105 qcastatemachine::SubscriptionQCaStateMachine Class - Reference

Inheritance diagram for qcastatemachine::SubscriptionQCaStateMachine:



Public Member Functions

- **SubscriptionQCaStateMachine** (void *parent)
- bool **process** (int requestedState)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QCaStateMachine.h
- /home/andrew/epicsqt/framework/data/src/QCaStateMachine.cpp

9.106 trace Class Reference

Public Attributes

- QVector< QCaDateTime > **timeStamps**
- QVector< double > **xdata**
- QVector< double > **ydata**
- QwtPlotCurve * **curve**
- QColor **color**
- QString **legend**
- bool **waveform**
- QwtPlotCurve::CurveStyle **style**

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEPlot/QEPlot.h

9.107 TrackRange Class Reference

Public Member Functions

- void **clear** ()
- void **merge** (const double d)
- void **merge** (const TrackRange that)
- bool **getMinMax** (double &min, double &max)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChartItem.h
- /home/andrew/epicsqt/framework/widgets/QEStripChart/QEStripChartItem.cpp

9.108 userInfoStruct Class Reference

Public Attributes

- bool **enable**
- double **value1**
- double **value2**
- QString **elementText**

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEPeriodic/QEPeriodic.h

9.109 QEPeriodic::userInfoStructArray Struct Reference

Public Attributes

- userInfoStruct **array** [NUM_ELEMENTS]

The documentation for this struct was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEPeriodic/QEPeriodic.h

9.110 userLevelSignal Class Reference

Signals

- void userChanged (userLevels level)

  Internal use only. Send when the user level has changed.

Public Member Functions

- void **setLevel** (userLevels levelIn)
- userLevels **getLevel** ()

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/ContainerProfile.h
- /home/andrew/epicsqt/framework/widgets/src/ContainerProfile.cpp

9.111 userLevelSlot Class Reference

Public Slots

- void **userChanged** (userLevels level)

Public Member Functions

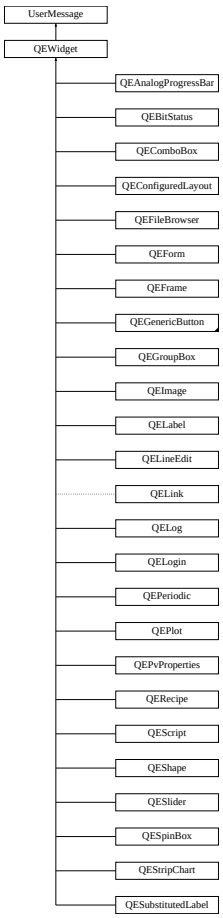
- void **setOwner** (ContainerProfile *ownerIn)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/ContainerProfile.h
- /home/andrew/epicsqt/framework/widgets/src/ContainerProfile.cpp

9.112 UserMessage Class Reference

Inheritance diagram for UserMessage:



Public Types

- enum **message_filter_options** { **MESSAGE_FILTER_ANY**, **MESSAGE_FILTER_MATCH**, **MESSAGE_FILTER_NONE** }

Public Member Functions

- void **setSourceId** (unsigned int sourceId)
- void **setFormId** (unsigned int formId)
- void **setFormFilter** (message_filter_options formFilterIn)
- void **setSourceFilter** (message_filter_options sourceFilterIn)
- unsigned int **getSourceId** ()
- unsigned int **getFormId** ()
- message_filter_options **getFormFilter** ()
- message_filter_options **getSourceFilter** ()
- void **setChildFormId** (unsigned int)
- unsigned int **getChildFormId** ()

- unsigned int **getNextMessageFormId** ()
- void **sendMessage** (QString message, message_types type=MESSAGE_TYPE_INFO)
- void **sendMessage** (QString message, QString source, message_types type=MESSAGE_TYPE_INFO)
- QString **getMessageTypeName** (message_types type)
- virtual void **newMessage** (QString, message_types)

Friends

- class **UserMessageSlot**
- class **UserMessageSignal**

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/UserMessage.h
- /home/andrew/epicsqt/framework/widgets/src/UserMessage.cpp

9.113 UserMessageSignal Class Reference

Signals

- void **message** (QString msg, message_types type, unsigned int formId, unsigned int sourceId, UserMessage *originator)

Public Member Functions

- void **sendMessage** (QString msg, message_types type, unsigned int formId, unsigned int sourceId, UserMessage *originator)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/UserMessage.h
- /home/andrew/epicsqt/framework/widgets/src/UserMessage.cpp

9.114 UserMessageSlot Class Reference

Public Slots

- void **message** (QString msg, message_types type, unsigned int formId, unsigned int sourceId, UserMessage *originator)

Public Member Functions

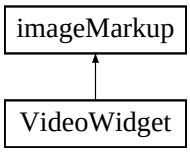
- void **setOwner** (UserMessage ∗ownerIn)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/include/UserMessage.h
- /home/andrew/epicsqt/framework/widgets/src/UserMessage.cpp

9.115 VideoWidget Class Reference

Inheritance diagram for VideoWidget:



Signals

- void **userSelection** (imageMarkup::markupIds mode, QPoint point1, QPoint point2)
- void **zoomInOut** (int zoomAmount)
- void **currentPixelInfo** (QPoint pos)
- void **pan** (QPoint pos)

Public Member Functions

- **VideoWidget** (QWidget ∗parent=0)
- void **setNewImage** (const QImage image, QCaDateTime &time)
- void **setPanning** (bool panningIn)
- bool **getPanning** ()
- QPoint **scalePoint** (QPoint pnt)
- int **scaleOrdinate** (int ord)
- QImage **getImage** ()

Protected Member Functions

- void **paintEvent** (QPaintEvent ∗)
- void **mousePressEvent** (QMouseEvent ∗event)
- void **mouseReleaseEvent** (QMouseEvent ∗event)
- void **mouseMoveEvent** (QMouseEvent ∗event)

- void **wheelEvent** (QWheelEvent ∗event)
- void **markupChange** (QImage &markups, QVector< QRect > &changedAreas)
- void **resizeEvent** (QResizeEvent ∗event)
- void **markupSetCursor** (QCursor cursor)
- void **markupAction** (markupIds mode, QPoint point1, QPoint point2)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/videowidget.h
- /home/andrew/epicsqt/framework/widgets/QEImage/videowidget.cpp

9.116 QEPvProperties::WidgetHolder Struct Reference

Public Attributes

- QVBoxLayout ∗ **layout**
- QComboBox ∗ **box**
- QLabel ∗ **timeStamp**
- QTableWidget ∗ **table**

The documentation for this struct was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/QEPvProperties/QEPvProperties.h

9.117 WidgetRef Class Reference

Public Member Functions

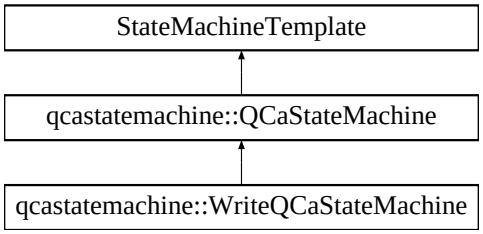
- **WidgetRef** (QEWidget ∗refIn)
- QEWidget ∗ **getRef** ()

The documentation for this class was generated from the following file:

- /home/andrew/epicsqt/framework/widgets/include/ContainerProfile.h

9.118 qcastatemachine::WriteQCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::WriteQCaStateMachine:



Public Member Functions

- **WriteQCaStateMachine** (void ∗parent)
- bool **process** (int requestedState)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/data/include/QCaStateMachine.h
- /home/andrew/epicsqt/framework/data/src/QCaStateMachine.cpp

9.119 zoomMenu Class Reference

Public Member Functions

- **zoomMenu** (QWidget ∗parent=0)
- void **enableAreaSelected** (bool enable)
- imageContextMenu::imageContextMenuOptions **getZoom** (const QPoint &pos)

The documentation for this class was generated from the following files:

- /home/andrew/epicsqt/framework/widgets/QEImage/zoomMenu.h
- /home/andrew/epicsqt/framework/widgets/QEImage/zoomMenu.cpp