EPICS QT Framework

2.1

Generated by Doxygen 1.7.5

Tue Jan 8 2013 20:05:48

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
```

Alternativly, get a packaged file (epicsqt.tar.gz) from the epicsqt repository site.

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:

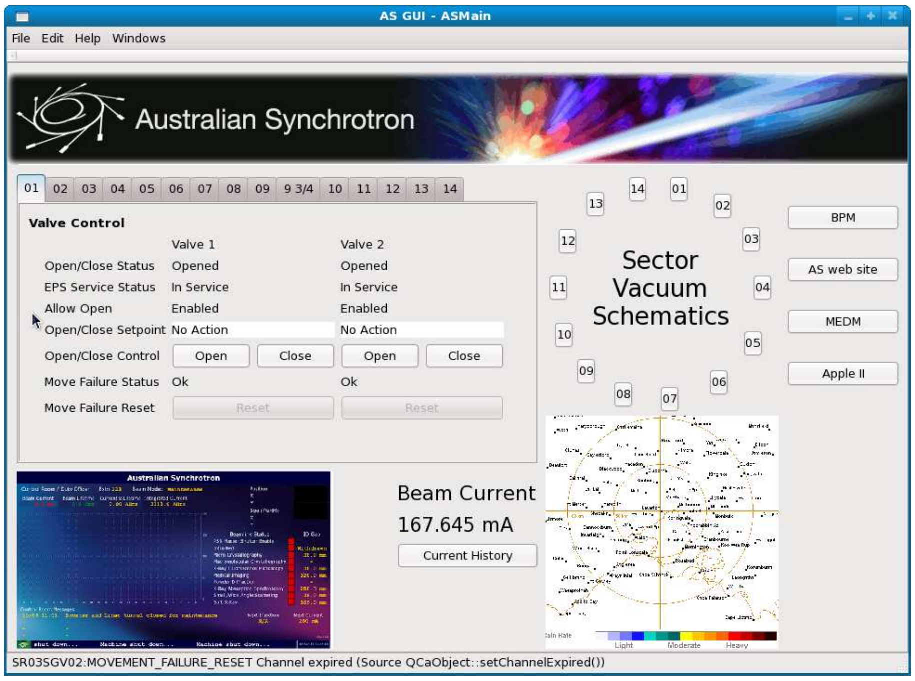
Andrew Rhyder <`andrew.rhyder@synchrotron.org.au`>

Chapter 2

GNU General Public License

Chapter 3

ASgui screen shots



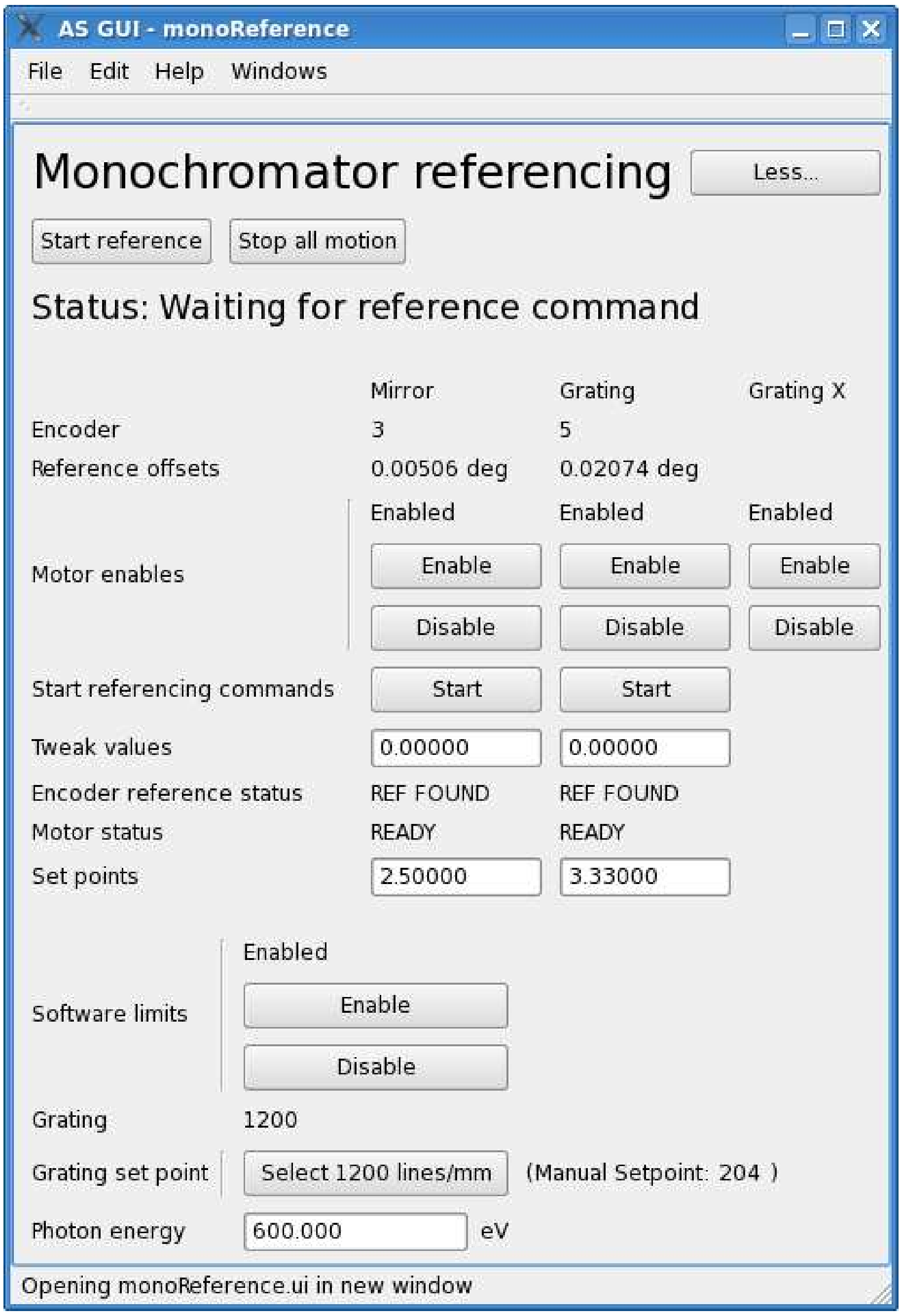
Valve Control

	Valve 1		Valve 2	
Open/Close Status	Opened		Opened	
EPS Service Status	In Service		In Service	
Allow Open	Enabled		Enabled	
Open/Close Setpoint	No Action		No Action	
Open/Close Control	Open	Close	Open	Close
Move Failure Status	Ok		Ok	
Move Failure Reset	Reset		Reset	

Beam Current
167.645 mA

SR03SGV02:MOVEMENT_FAILURE_RESET Channel expired (Source QCaObject::setChannelExpired())

Figure 3.1: Australian Synchrotron mock up



AS GUI - monoReference

File Edit Help Windows

Monochromator referencing

Less...

Start reference

Stop all motion

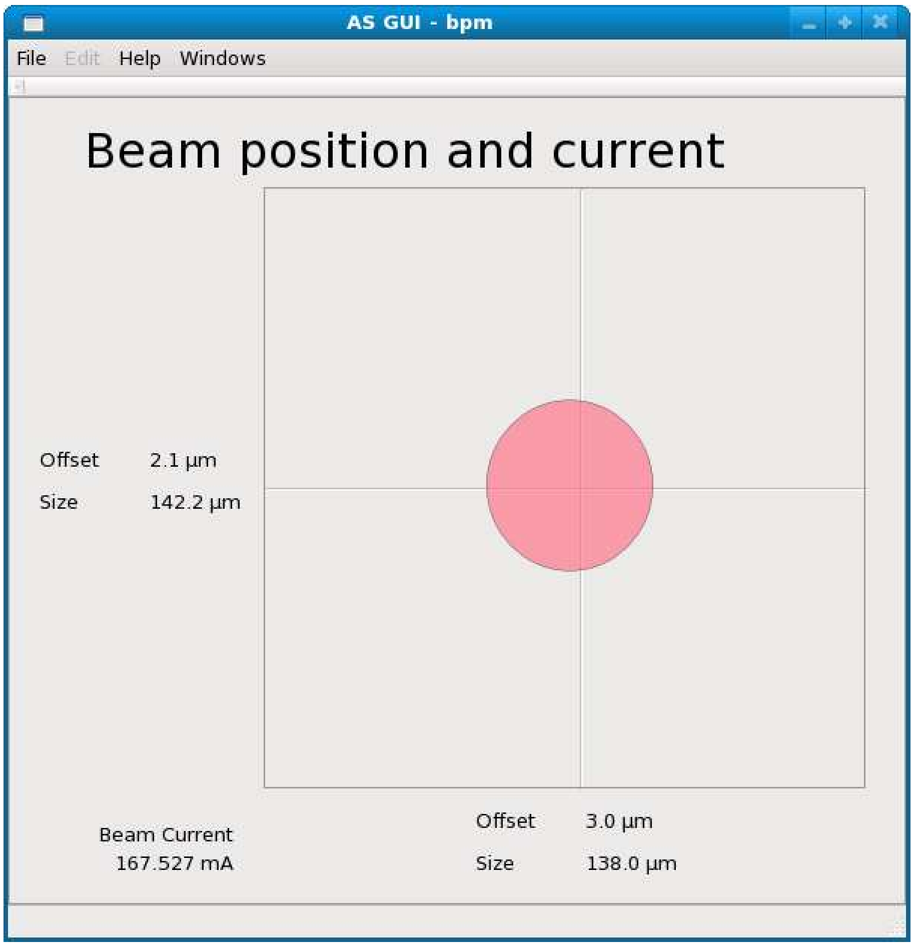
Status: Waiting for reference command

	Mirror	Grating	Grating X
Encoder	3	5	
Reference offsets	0.00506 deg	0.02074 deg	
Motor enables	Enabled	Enabled	Enabled
	Enable	Enable	Enable
	Disable	Disable	Disable
Start referencing commands	Start	Start	
Tweak values	0.00000	0.00000	
Encoder reference status	REF FOUND	REF FOUND	
Motor status	READY	READY	
Set points	2.50000	3.33000	

Software limits	Enabled
	Enable
	Disable
Grating	1200
Grating set point	Select 1200 lines/mm (Manual Setpoint: 204)
Photon energy	600.000 eV

Opening monoReference.ui in new window

Figure 3.2: Monochromator referencing



Beam position and current

Offset 2.1 µm

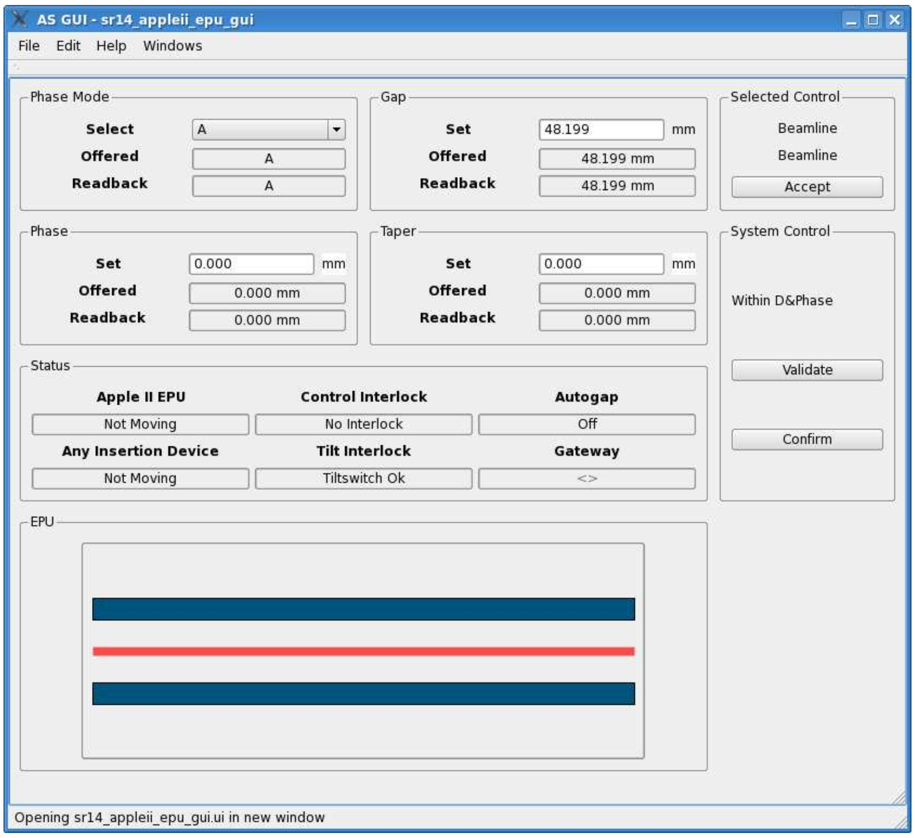
Size 142.2 µm

Beam Current
167.527 mA

Offset 3.0 µm

Size 138.0 µm

Figure 3.3: Beam position monitor



Select	A	Set	48.199	mm
Offered	A	Offered	48.199 mm	
Readback	A	Readback	48.199 mm	

Selected Control: Beamline, Beamline

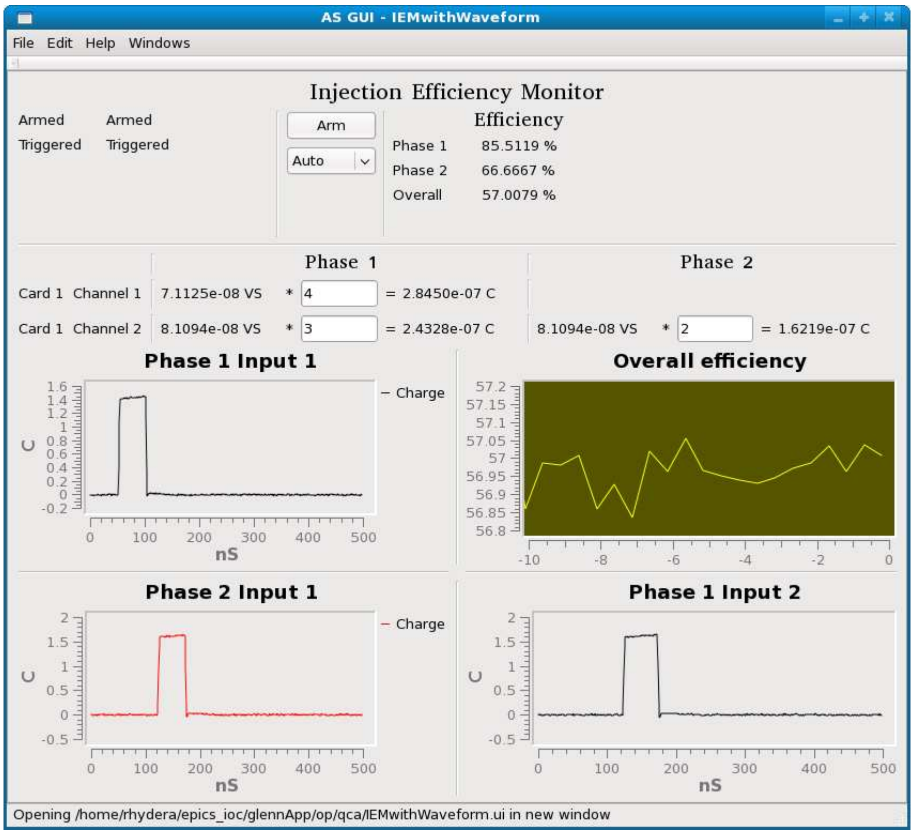
Set	0.000	mm	Set	0.000	mm
Offered	0.000 mm		Offered	0.000 mm	
Readback	0.000 mm		Readback	0.000 mm	

System Control: Within D&Phase

Apple II EPU	Control Interlock	Autogap
Not Moving	No Interlock	Off
Any Insertion Device	**Tilt Interlock**	**Gateway**
Not Moving	Tiltswitch Ok	<>

Opening sr14_appleii_epu_gui.ui in new window

Figure 3.4: Insertion device



Injection Efficiency Monitor

Armed Armed

Triggered Triggered

Auto

Efficiency	
Phase 1	85.5119 %
Phase 2	66.6667 %
Overall	57.0079 %

	Phase 1				Phase 2			
Card 1 Channel 1	7.1125e-08 VS	*	4	= 2.8450e-07 C				
Card 1 Channel 2	8.1094e-08 VS	*	3	= 2.4328e-07 C	8.1094e-08 VS	*	2	= 1.6219e-07 C

Opening /home/rhydera/epics_ioc/glennApp/op/qca/IEMwithWaveform.ui in new window

Figure 3.5: Injection efficiency monitor

Chapter 4

other applications using epicsqt widgets

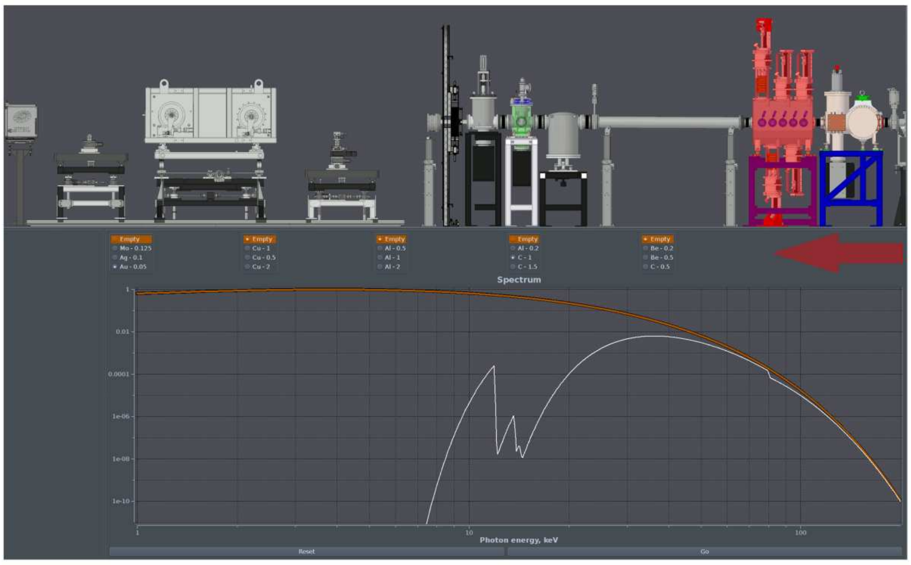
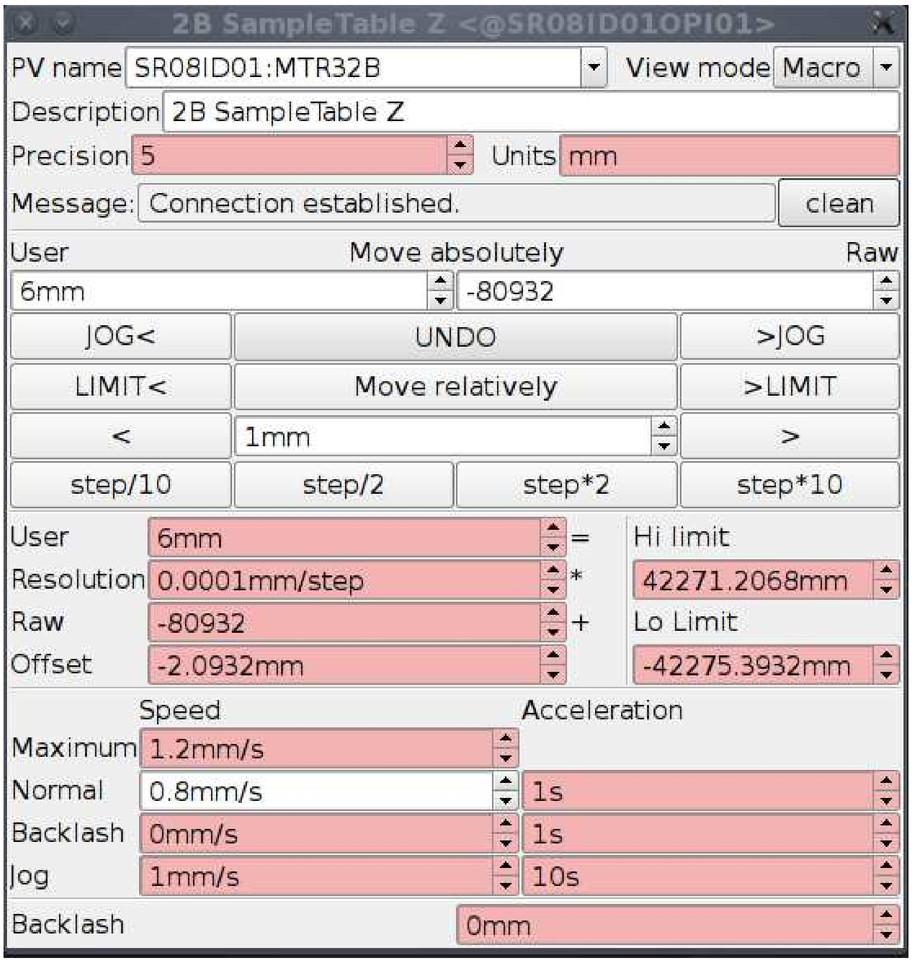


Figure 4.1: Medical Imaging beamline


2B SampleTable Z <@SR08ID01OPI01>

PV name: SR08ID01:MTR32B
View mode: Macro
Description: 2B SampleTable Z
Precision: 5
Units: mm
Message: Connection established.

User	Move absolutely	Raw
6mm		-80932

JOG<	UNDO	>JOG
LIMIT<	Move relatively	>LIMIT
<	1mm	>

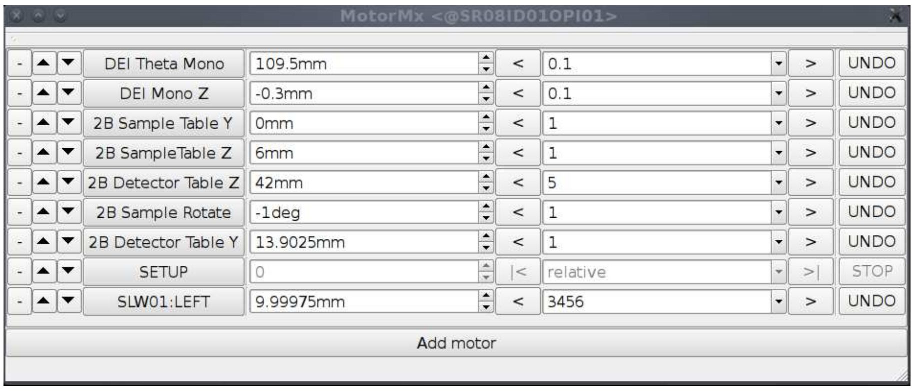
step/10	step/2	step*2	step*10

User	6mm	=	Hi limit
Resolution	0.0001mm/step	*	42271.2068mm
Raw	-80932	+	Lo Limit
Offset	-2.0932mm		-42275.3932mm

	Speed	Acceleration
Maximum	1.2mm/s	
Normal	0.8mm/s	1s
Backlash	0mm/s	1s
Jog	1mm/s	10s

Backlash: 0mm

Figure 4.2: Motor controller


MotorMx <@SR08ID01OPI01>

-	▲	▼	DEI Theta Mono	109.5mm	<	0.1	>	UNDO
-	▲	▼	DEI Mono Z	-0.3mm	<	0.1	>	UNDO
-	▲	▼	2B Sample Table Y	0mm	<	1	>	UNDO
-	▲	▼	2B SampleTable Z	6mm	<	1	>	UNDO
-	▲	▼	2B Detector Table Z	42mm	<	5	>	UNDO
-	▲	▼	2B Sample Rotate	-1deg	<	1	>	UNDO
-	▲	▼	2B Detector Table Y	13.9025mm	<	1	>	UNDO
-	▲	▼	SETUP	0	|<	relative	>|	STOP
-	▲	▼	SLW01:LEFT	9.99975mm	<	3456	>	UNDO

Add motor

Figure 4.3: Motor controller

Chapter 5

Qt Designer

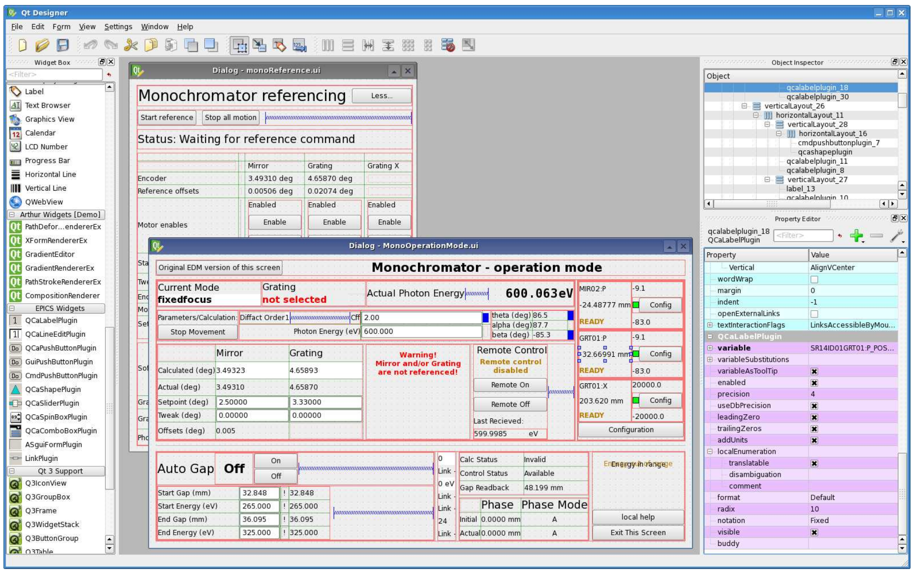
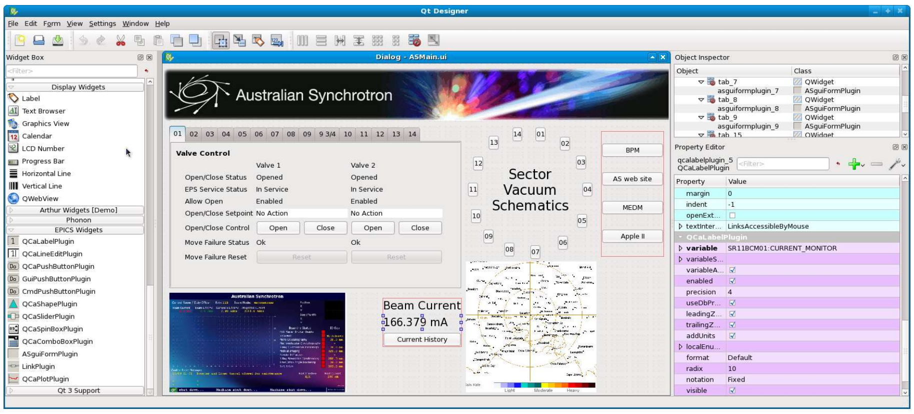

Qt Designer
File Edit Form View Settings Window Help
Widget Box
Dialog - monoReference.ui
Monochromator referencing
Status: Waiting for reference command
Dialog - MonoOperationMode.ui
Monochromator - operation mode
Auto Gap Off
Object Inspector
Property Editor

Figure 5.1: Editing multiple GUIs



Valve Control

	Valve 1	Valve 2
Open/Close Status	Opened	Opened
EPS Service Status	In Service	In Service
Allow Open	Enabled	Enabled
Open/Close Setpoint	No Action	No Action
Open/Close Control	Open Close	Open Close
Move Failure Status	Ok	Ok
Move Failure Reset	Reset	Reset

Object Inspector

Object	Class
tab_7	QWidget
asguiformplugin_7	ASguiFormPlugin
tab_8	QWidget
asguiformplugin_8	ASguiFormPlugin
tab_9	QWidget
asguiformplugin_9	ASguiFormPlugin
tab_15	QWidget

Property Editor

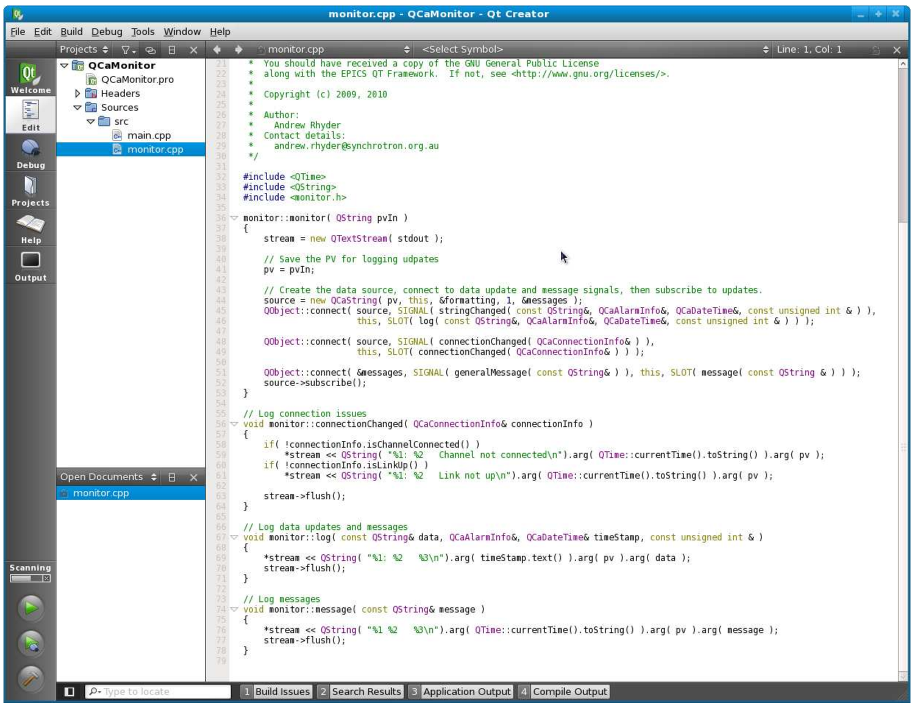
qcalabelplugin_5
QCaLabelPlugin

Property	Value
margin	0
indent	-1
openExt...	☐
textInter...	LinksAccessibleByMouse
QCaLabelPlugin	
variable	SR11BCM01:CURRENT_MONITOR
variableS...	
variableA...	✓
enabled	✓
precision	4
useDbPr...	✓
leadingZ...	✓
trailingZ...	✓
addUnits	✓
localEnu...	
format	Default
radix	10
notation	Fixed
visible	✓

Figure 5.2: Editing a GUI

Chapter 6

Qt Creator



```
 *  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/>.
 *
 *  Copyright (c) 2009, 2010
 *
 *  Author:
 *    Andrew Rhyder
 *  Contact details:
 *    andrew.rhyder@synchrotron.org.au
 */

#include <QTime>
#include <QString>
#include <monitor.h>

monitor::monitor( QString pvIn )
{
    stream = new QTextStream( stdout );

    // Save the PV for logging udpates
    pv = pvIn;

    // Create the data source, connect to data update and message signals, then subscribe to updates.
    source = new QCaString( pv, this, &formatting, 1, &messages );
    QObject::connect( source, SIGNAL( stringChanged( const QString&, QCaAlarmInfo&, QCaDateTime&, const unsigned int & ) ),
                      this, SLOT( log( const QString&, QCaAlarmInfo&, QCaDateTime&, const unsigned int & ) ) );

    QObject::connect( source, SIGNAL( connectionChanged( QCaConnectionInfo& ) ),
                      this, SLOT( connectionChanged( QCaConnectionInfo& ) ) );

    QObject::connect( &messages, SIGNAL( generalMessage( const QString& ) ), this, SLOT( message( const QString & ) ) );
    source->subscribe();
}

// Log connection issues
void monitor::connectionChanged( QCaConnectionInfo& connectionInfo )
{
    if( !connectionInfo.isChannelConnected() )
        *stream << QString( "%1: %2   Channel not connected\n").arg( QTime::currentTime().toString() ).arg( pv );
    if( !connectionInfo.isLinkUp() )
        *stream << QString( "%1: %2   Link not up\n").arg( QTime::currentTime().toString() ).arg( pv );

    stream->flush();
}

// Log data updates and messages
void monitor::log( const QString& data, QCaAlarmInfo&, QCaDateTime& timeStamp, const unsigned int & )
{
    *stream << QString( "%1: %2   %3\n").arg( timeStamp.text() ).arg( pv ).arg( data );
    stream->flush();
}

// Log messages
void monitor::message( const QString& message )
{
    *stream << QString( "%1 %2   %3\n").arg( QTime::currentTime().toString() ).arg( pv ).arg( message );
    stream->flush();
}
```

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 p-F=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 p-GroupName="", 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

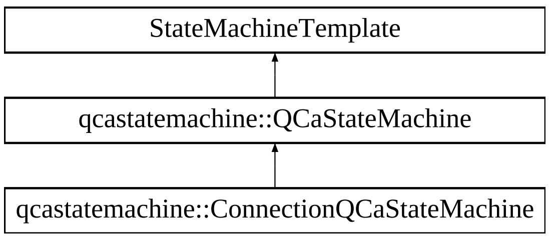
9.10 QEAnalogIndicator::BandList Class Reference

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:


StateMachineTemplate
qcastatemachine::QCaStateMachine
qcastatemachine::ConnectionQCaStateMachine

Public Member Functions

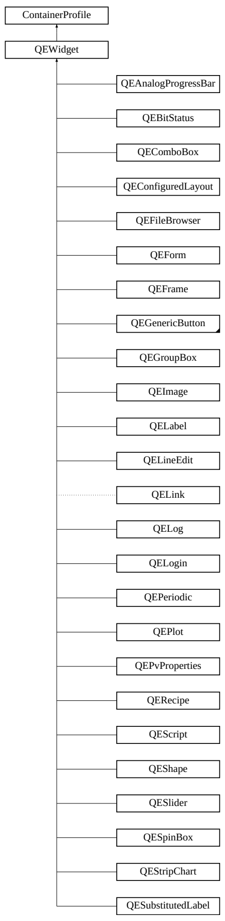
- **ConnectionQCaStateMachine** (void ∗parent)
- bool **process** (int requestedState)

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)

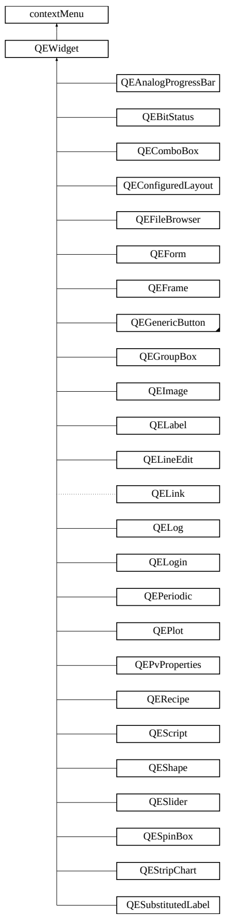
- void **updateConsumers** (QObject ∗guiLaunchConsumerIn)
- QObject ∗ **replaceGuiLaunchConsumer** (QObject ∗newGuiLaunchConsumer-In)
- void **addMacroSubstitutions** (QString macroSubstitutionsIn)
- void **removeMacroSubstitutions** ()
- QObject ∗ **getGuiLaunchConsumer** ()
- QString **getPath** ()
- 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_SPECIF-IC_WIDGETS_START_HERE, **ICM_SAVE**, **ICM_PAUSE**, **ICM_ENABLE_TIME**, **ICM_ENABLE_CURSOR_PIXEL**, **ICM_ENABLE_CONTRAST_REVERSAL**, **I-CM_ENABLE_VERT**, **ICM_ENABLE_HOZ**, **ICM_ENABLE_AREA**, **ICM_ENA-BLE_LINE**, **ICM_ENABLE_TARGET**, **ICM_DISPLAY_BUTTON_BAR**, **ICM_Z-OOM_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_ROTA-TE_RIGHT**, **ICM_ROTATE_LEFT**, **ICM_ROTATE_180**, **ICM_FLIP_HORIZONT-AL**, **ICM_FLIP_VERTICAL**, **ICM_SELECT_PAN**, **ICM_SELECT_HSLICE**, **ICM-_SELECT_VSLICE**, **ICM_SELECT_AREA**, **ICM_SELECT_PROFILE**, **ICM_SE-LECT_TARGET**, **ICM_SELECT_BEAM**, **ICM_CLEAR_MARKUP** }

Public Member Functions

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

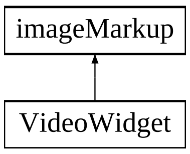
- void **addMenuItem** (const QString &title, const bool checkable, const bool checked, const imageContextMenuOptions option)
- void **addOptionMenuItem** (const QString &title, const bool checkable, const bool checked, const imageContextMenuOptions option)

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 **setShowTime** (bool visibleIn)
- bool **getShowTime** ()
- markupIds **getMode** ()
- void **setMode** (markupIds modeIn)
- void **setMarkupColor** (markupIds mode, QColor markupColorIn)
- QColor **getMarkupColor** (markupIds mode)
- bool **showMarkupMenu** (const QPoint &pos, const QPoint &globalPos)
- QCursor **getCircleCursor** ()
- QCursor **getTargetCursor** ()
- virtual void **markupSetCursor** (QCursor cursor)=0

Public Attributes

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

Protected Member Functions

- bool **anyVisibleMarkups** ()
- QVector< QRect > & **getMarkupAreas** ()
- QCursor **getDefaultMarkupCursor** ()
- void **setMarkupTime** (QCaDateTime &time)
- bool **markupMousePressEvent** (QMouseEvent ∗event, bool panning)
- bool **markupMouseReleaseEvent** (QMouseEvent ∗event, bool panning)
- bool **markupMouseMoveEvent** (QMouseEvent ∗event, bool panning)
- void **markupResize** (QSize newSize)
- virtual void **markupChange** (QImage &markups, QVector< QRect > &changed-Areas)=0
- virtual void **markupAction** (markupIds mode, bool clearing, QPoint point1, Q-Point 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

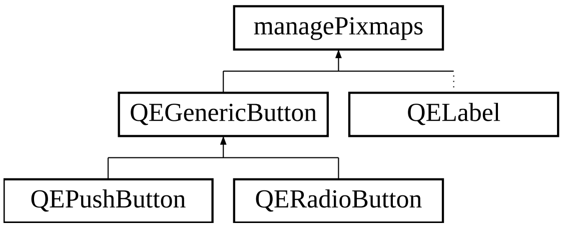
- double **dValue**
- QString **sValue**
- operations **op**
- QString **text**

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

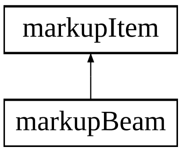
- void **setDataPixmap** (const QPixmap &Pixmap, const unsigned int index)
- QPixmap **getDataPixmap** (const unsigned int index)
- QPixmap **getDataPixmap** (const QString value)

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** ()

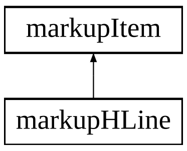
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()
- void **scaleSpecific** (double xScale, double yScale)

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)

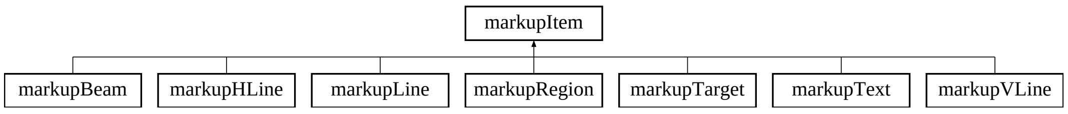
- void **startDrawing** (QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (QPoint pos)
- bool **isOver** (QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()
- void **scaleSpecific**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

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 Member Functions

- void **erase** ()
- void **drawMarkupIn** ()
- void **drawMarkupOut** ()
- void **setColor** (QColor colorIn)
- void **scale** (double xScale, double yScale)
- virtual QPoint **origin** ()=0
- virtual void **moveTo** (QPoint pos)=0
- virtual void **startDrawing** (QPoint pos)=0
- virtual bool **isOver** (const QPoint point, QCursor ∗cursor)=0
- virtual QPoint **getPoint1** ()=0
- virtual QPoint **getPoint2** ()=0
- virtual QCursor **defaultCursor** ()=0

Public Attributes

- QRect **area**
- bool **visible**
- bool **interactive**
- bool **reportOnMove**
- QColor **color**

Protected Types

- enum **isOverOptions** { **OVER_LINE**, **OVER_BORDER**, **OVER_AREA** }
- enum **markupHandles** { **MARKUP_HANDLE_NONE**, **MARKUP_HANDLE_S-TART**, **MARKUP_HANDLE_END**, **MARKUP_HANDLE_TL**, **MARKUP_HAND-LE_TR**, **MARKUP_HANDLE_BL**, **MARKUP_HANDLE_BR**, **MARKUP_HAND-LE_T**, **MARKUP_HANDLE_B**, **MARKUP_HANDLE_L**, **MARKUP_HANDLE_R** }

Protected Member Functions

- **markupItem** (imageMarkup ∗ownerIn, isOverOptions over, bool interactiveIn, bool reportOnMoveIn)
- virtual void **setArea** ()=0
- virtual void **drawMarkup** (QPainter &p)=0
- bool **pointIsNear** (QPoint p1, QPoint p)
- QColor **getColor** ()

Protected Attributes

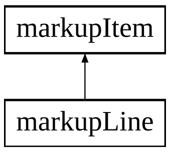
- markupHandles **activeHandle**
- isOverOptions **isOverType**
- bool **highlighted**
- int **highlightMargin**
- imageMarkup ∗ **owner**

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** ()

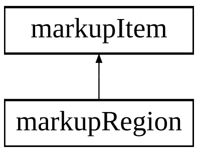
- void **scaleSpecific** (double xScale, double yScale)

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

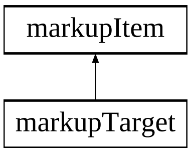
- **markupRegion** (imageMarkup ∗ownerIn, bool interactiveIn, bool reportOnMoveIn)
- void **startDrawing** (QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (QPoint pos)
- bool **isOver** (QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QPoint **getPoint1**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

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

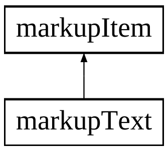
- **markupTarget** (imageMarkup ∗ownerIn, bool interactiveIn, bool reportOnMoveIn)
- void **startDrawing** (QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (QPoint pos)
- bool **isOver** (QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QPoint **getPoint1**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

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)

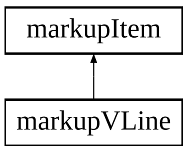
- void **moveTo** (QPoint pos)
- bool **isOver** (QPoint point, QCursor ∗cursor)
- QPoint **origin** ()
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()
- void **scaleSpecific** (double xScale, double yScale)

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** ()
- void **scaleSpecific** (double xScale, double yScale)

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$<$ Q-String $>$ &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/PeriodicElementSetup-Form.h
- /home/andrew/epicsqt/framework/widgets/QEPeriodic/PeriodicElementSetup-Form.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 **setProfile** (QVector< QPointF > &profile, double minX, double maxX, double minY, double maxY)

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

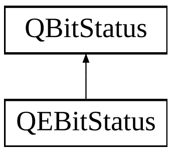
- int **width**
- const QString **caption**
- const QString **iconName**
- const QString **toolTip**
- const char ∗ **member**

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 linkState-In)

- 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

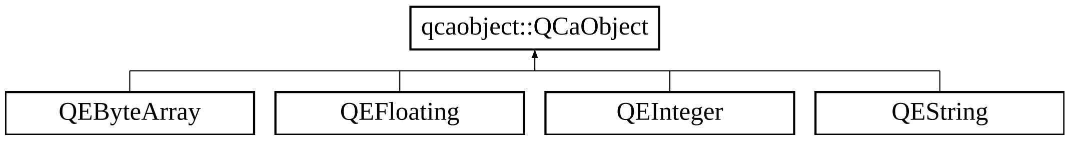
- QObject ∗ **eventObject**
- long **referenceCount**

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

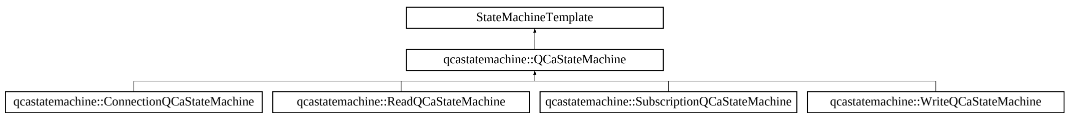
- static void **processEventStatic** (QCaEventUpdate ∗dataUpdateEvent)

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:


StateMachineTemplate
qcastatemachine::QCaStateMachine
qcastatemachine::ConnectionQCaStateMachine
qcastatemachine::ReadQCaStateMachine
qcastatemachine::SubscriptionQCaStateMachine
qcastatemachine::WriteQCaStateMachine

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)

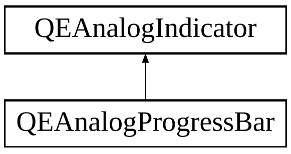
The documentation for this class was generated from the following files:

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

9.49 QEAnalogIndicator Class Reference

`#include <QEAnalogIndicator.h>`

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)

  Constructor.
- virtual ∼QEAnalogIndicator ()

  Destructor.
- virtual QSize sizeHint () const

  Size hint.
- double getValue ()

  Access function for value property - refer to value property for details.
- void setMinimum (const double value)

  Access function for minimum - refer to minimum property for details.
- double getMinimum ()

  Access function for minimum - refer to minimum property for details.
- void setMaximum (const double value)

  Access function for maximum - refer to maximum property for details.
- double getMaximum ()

  Access function for maximum - refer to maximum property for details.
- void setOrientation (const enum Orientations value)

  Access function for orientation - refer to orientation property for details.
- enum Orientations getOrientation ()

  Access function for orientation - refer to orientation property for details.
- void setMode (const enum Modes value)

  Access function for mode - refer to mode property for details.
- enum Modes getMode ()

  Access function for mode - refer to mode property for details.
- void setCentreAngle (const int value)

  Access function for centreAngle - refer to centreAngle property for details.
- int getCentreAngle ()

  Access function for centreAngle - refer to centreAngle property for details.
- void setSpanAngle (const int value)

  Access function for spanAngle - refer to spanAngle property for details.
- int getSpanAngle ()

  Access function for spanAngle - refer to spanAngle property for details.
- void setMinorInterval (const double value)

  Access function for minorInterval - refer to minorInterval property for details.
- double getMinorInterval ()

  Access function for minorInterval - refer to minorInterval property for details.
- void setMajorInterval (const double value)

  Access function for majorInterval - refer to majorInterval property for details.
- double getMajorInterval ()

  Access function for majorInterval - refer to majorInterval property for details.
- void setLogScaleInterval (const int value)

Access function for logScaleInterval - refer to logScaleInterval property for details.

- int getLogScaleInterval ()

  Access function for logScaleInterval - refer to logScaleInterval property for details.
- void setBorderColour (const QColor value)

  Access function for borderColour - refer to borderColour property for details.
- QColor getBorderColour ()

  Access function for borderColour - refer to borderColour property for details.
- void setForegroundColour (const QColor value)

  Access function for foregroundColour - refer to foregroundColour property for details.
- QColor getForegroundColour ()

  Access function for foregroundColour - refer to foregroundColour property for details.
- void setBackgroundColour (const QColor value)

  Access function for backgroundColour - refer to backgroundColour property for details.
- QColor getBackgroundColour ()

  Access function for backgroundColour - refer to backgroundColour property for details.
- void setFontColour (const QColor value)

  Access function for fontColour - refer to fontColour property for details.
- QColor getFontColour ()

  Access function for fontColour - refer to fontColour property for details.
- void setShowText (const bool value)

  Access function for showText - refer to showText property for details.
- bool getShowText ()

  Access function for showText - refer to showText property for details.
- void setShowScale (const bool value)

  Access function for showScale - refer to showScale property for details.
- bool getShowScale ()

  Access function for showScale - refer to showScale property for details.
- void setLogScale (const bool value)

  Access function for logScale - refer to logScale property for details.
- bool getLogScale ()

  Access function for logScale - refer to logScale property for details.

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

9.49.1 Detailed Description

This class provides a non CA aware graphical analog indicator base class. It supports a number of display modes including Bar, Scale and Meter.

When in Bar mode, it mimics QProgressBar and provides an analog progress bar widget.

9.49.2 Member Enumeration Documentation

9.49.2.1 enum QEAnalogIndicator::Modes

The type of analog indicator used to represent the value

Enumerator:

Bar Bar (solid bar from minimum up to current value)

Scale Scale (diamond marker tracks current value)

Meter Meter (Needle moving across an arc scale)

9.49.2.2 enum QEAnalogIndicator::Orientations

The orientation of Bar and Scale indicators

Enumerator:

Left_To_Right Left to right.

Top_To_Bottom Top to bottom.

Right_To_Left Right to left.

Bottom_To_Top Bottom to top.

9.49.3 Property Documentation

9.49.3.1 QColor QEAnalogIndicator::backgroundColour `[read, write]`

Background colour

9.49.3.2 QColor QEAnalogIndicator::borderColour `[read, write]`

Border colour

9.49.3.3 int QEAnalogIndicator::centreAngle `[read, write]`

The angle in degreed of the line that Meter indicators are centered around. Zero represents a vertical centerline and angles increment clockwise.

9.49.3.4 QColor QEAnalogIndicator::fontColour `[read, write]`

Font colour

9.49.3.5 QColor QEAnalogIndicator::foregroundColour `[read, write]`

Foreground colour

9.49.3.6 bool QEAnalogIndicator::logScale `[read, write]`

If set, use a logarithmic scale. If clear, use a linear scale

9.49.3.7 int QEAnalogIndicator::logScaleInterval `[read, write]`

Log scale interval.

9.49.3.8 double QEAnalogIndicator::majorInterval `[read, write]`

Minor scale interval. Only applies for linear scale (not log scale)

9.49.3.9 double QEAnalogIndicator::maximum `[read, write]`

Maximum indicated value.

9.49.3.10 double QEAnalogIndicator::minimum `[read, write]`

Minimum indicated value.

9.49.3.11 double QEAnalogIndicator::minorInterval `[read, write]`

Minor scale interval. Only applies for linear scale (not log scale)

9.49.3.12 Modes QEAnalogIndicator::mode `[read, write]`

Selects what type of indicator is used (refer to Modes)

9.49.3.13 Orientations QEAnalogIndicator::orientation `[read, write]`

The orientation of Bar and Scale indicators (refer to Orientations)

9.49.3.14 bool QEAnalogIndicator::showScale `[read, write]`

If set, show the scale

9.49.3.15 bool QEAnalogIndicator::showText `[read, write]`

If set, show textual representation of value on the indicator

9.49.3.16 int QEAnalogIndicator::spanAngle `[read, write]`

The span of the Meter scale arc in degrees Typical meters are 180 deg and 270 deg

9.49.3.17 double QEAnalogIndicator::value `[read, write]`

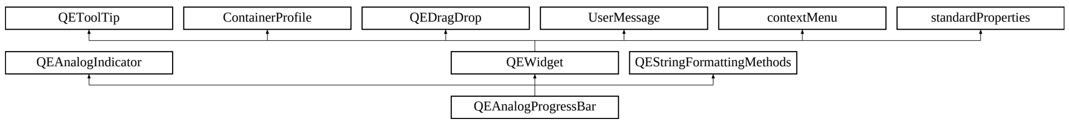
Current indicated value.

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 }
- 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 ArrayActions { Append = QEStringFormatting::APPEND, Ascii = QEStringFormatting::ASCII, Index = QEStringFormatting::INDEX }
- 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)
- virtual ∼QEAnalogProgressBar ()

  Destruction.
- void setVariableNameAndSubstitutions (QString variableNameIn, QString variableNameSubstitutionsIn, unsigned int variableIndex)
- void setUseDbDisplayLimits (bool useDbDisplayLimitsIn)

  Access function for useDbDisplayLimits property - refer to useDbDisplayLimits property for details.
- bool getUseDbDisplayLimits ()

  Access function for useDbDisplayLimits property - refer to useDbDisplayLimits property for details.
- void setAlarmSeverityDisplayMode (AlarmSeverityDisplayModes value)

  Access function for AlarmSeverityDisplayModes property - refer to AlarmSeverityDisplayModes property for details.
- AlarmSeverityDisplayModes getAlarmSeverityDisplayMode ()

  Access function for AlarmSeverityDisplayModes property - refer to AlarmSeverityDisplayModes property for details.

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 Enumeration Documentation

9.50.1.1 enum QEAnalogProgressBar::ArrayActions

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

Enumerator:

Append Refer to QEStringFormatting::APPEND for details.

Ascii Refer to QEStringFormatting::ASCII for details.

Index Refer to QEStringFormatting::INDEX for details.

9.50.1.2 enum QEAnalogProgressBar::Formats

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

Enumerator:

Default Format according to the EPICS database record type.

Floating Format as a floating point number.

Integer Format as an integer.

UnsignedInteger Format as an unsigned integer.

Time Format as a time.

LocalEnumeration Format as a selection from the localEnumeration property.

9.50.1.3 enum QEAnalogProgressBar::Notations

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

Enumerator:

Fixed Refer to QEStringFormatting::NOTATION_FIXED for details.

Scientific Refer to QEStringFormatting::NOTATION_SCIENTIFIC for details.

Automatic Refer to QEStringFormatting::NOTATION_AUTOMATIC for details.

9.50.1.4 enum QEAnalogProgressBar::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.50.2 Constructor & Destructor Documentation

9.50.2.1 QEAnalogProgressBar::QEAnalogProgressBar (QWidget ∗ *parent* = 0)

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

9.50.2.2 QEAnalogProgressBar::QEAnalogProgressBar (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.

9.50.3 Member Function Documentation

9.50.3.1 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.

9.50.3.2 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.

9.50.3.3 void QEAnalogProgressBar::setVariableNameAndSubstitutions (QString *variableNameIn,* QString *variableNameSubstitutionsIn,* unsigned int *variableIndex*) `[virtual]`

Virtual function that may be implimented by users of QEWidget to update variable names and macro substitutions. A default is provided that is suitible in most cases.

Reimplemented from QEWidget.

9.50.4 Property Documentation

9.50.4.1 bool QEAnalogProgressBar::addUnits `[read, write]`

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

9.50.4.2 AlarmSeverityDisplayModes QEAnalogProgressBar::alarmSeverityDisplayMode `[read, write]`

Visualise the EPICS alarm severity

9.50.4.3 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.

9.50.4.4 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’.

9.50.4.5 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.

9.50.4.6 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).

9.50.4.7 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.

9.50.4.8 bool QEAnalogProgressBar::leadingZero [read, write]

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

9.50.4.9 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.

Format is:

[[<|<=|=|!=|>=|>]value1|*] : string1 , [[<|<=|=|!=|>=|>]value2|*] : string2 , [[<|<=|=|!=|>=|>]value3|*] : string3 , ...

Where: < Less than <= Less than or equal = Equal (default if no operator specified) >= Greather than or equal > Greater than Always match (used to specify default text)

Values may be numeric or textual Values do not have to be in any order, but first match wins Values may be quoted Strings may be quoted Consecutive values do not have to be present. Operator is assumed to be equality if not present. White space is ignored except within quoted strings.

may be included in a string to indicate a line break

Examples are:

0:Off,1:On 0 : "Pump Running", 1 : "Pump not running" 0:"", 1:"Warning!\nAlarm" <2-:"Value is less than two", =2:"Value is equal to two", >2:"Value is grater than 2" 3:"-Beamline Available", *:"" "Pump Off":"OH NO!, the pump is OFF!","Pump On":"It's OK, the pump is on"

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:""'

A range of numbers can be covered by a pair of values as in the following example: >=4:"Between 4 and 8",<=8:"Between 4 and 8"

9.50.4.10 Notations QEAnalogProgressBar::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.50.4.11 int QEAnalogProgressBar::precision [read, write]

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

9.50.4.12 bool QEAnalogProgressBar::trailingZeros [read, write]

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

9.50.4.13 bool QEAnalogProgressBar::useDbDisplayLimits `[read, write]`

Use the EPICS database display limits

9.50.4.14 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.

9.50.4.15 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 wide 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'.

9.50.4.16 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.

9.50.4.17 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.

9.50.4.18 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.

9.50.4.19 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 wide through the QELogin widget, or programatically through setUser-Level() 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'.

9.50.4.20 QString QEAnalogProgressBar::variable [read, write]

EPICS variable name (CA PV)

9.50.4.21 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 QEToolTip.

9.50.4.22 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.

9.50.4.23 bool QEAnalogProgressBar::visible [read, write]

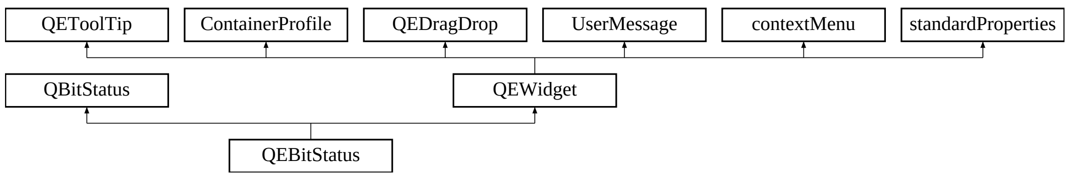
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

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

9.51 QEBitStatus Class Reference

Inheritance diagram for QEBitStatus:



Public Types

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

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 Enumeration Documentation

9.51.1.1 enum QEBitStatus::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.51.2 Member Function Documentation

9.51.2.1 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.

9.51.2.2 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.

9.51.2.3 void QEBitStatus::setVariableNameAndSubstitutions (QString *variableNameIn,* QString *variableNameSubstitutionsIn,* unsigned int *variableIndex*) `[virtual]`

Virtual function that may be implimented by users of QEWidget to update variable names and macro substitutions. A default is provided that is suitible in most cases.

Reimplemented from QEWidget.

9.51.3 Property Documentation

9.51.3.1 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.

9.51.3.2 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.

9.51.3.3 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.

9.51.3.4 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 wide 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'.

9.51.3.5 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 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.

9.51.3.6 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 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.

9.51.3.7 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.

9.51.3.8 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 wide through the QELogin widget, or programatically through setUser-Level() 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'.

9.51.3.9 QString QEBitStatus::variable `[read, write]`

EPICS variable name (CA PV)

9.51.3.10 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 QEToolTip.

9.51.3.11 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.

9.51.3.12 bool QEBitStatus::visible `[read, write]`

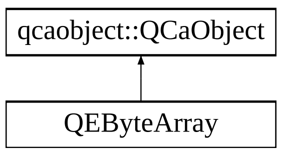
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

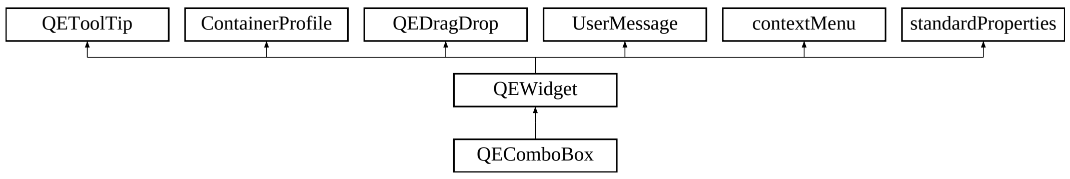
- **QEByteArray** (QString recordName, QObject ∗eventObject, unsigned int variableIndexIn)
- **QEByteArray** (QString recordName, QObject ∗eventObject, unsigned int variableIndexIn, UserMessage ∗userMessageIn)

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 }

Public Slots

- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const qlonglong &out)

- void userChange (const QString &oldValue, const QString &newValue, const Q-String &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 Enumeration Documentation

9.53.1.1 enum QEComboBox::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.53.2 Member Function Documentation

9.53.2.1 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.

9.53.2.2 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.

9.53.3 Member Data Documentation

9.53.3.1 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).

9.53.4 Property Documentation

9.53.4.1 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.

9.53.4.2 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.

9.53.4.3 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.

9.53.4.4 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.

9.53.4.5 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 wide 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'.

9.53.4.6 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 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.

9.53.4.7 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.

9.53.4.8 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.

9.53.4.9 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 wide through the QELogin widget, or programatically through setUser-Level() 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'.

9.53.4.10 QString QEComboBox::variable `[read, write]`

EPICS variable name (CA PV)

9.53.4.11 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 QEToolTip.

9.53.4.12 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.

9.53.4.13 bool QEComboBox::visible [read, write]

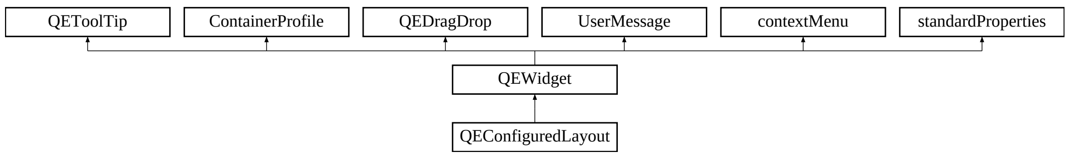
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

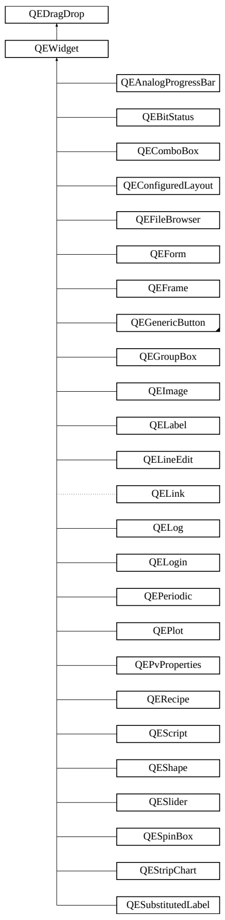
- **QEConfiguredLayoutManager** (QObject ∗pParent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- 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

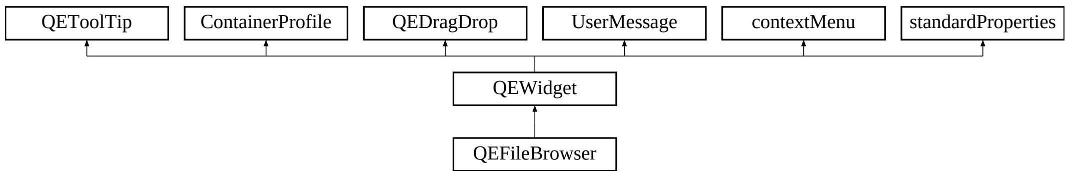
- void **qcaDragEnterEvent** (QDragEnterEvent ∗event)
- void **qcaDropEvent** (QDropEvent ∗event)
- void **qcaMousePressEvent** (QMouseEvent ∗event)
- virtual void **setDrop** (QVariant)
- virtual QVariant **getDrop** ()
- void **setAllowDrop** (bool allowDropIn)
- bool **getAllowDrop** ()

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

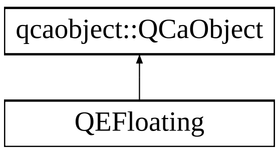
- QString **directoryPath**
- bool **showDirectoryPath**
- bool **showDirectoryBrowser**
- bool **showRefresh**
- bool **showColumnTime**
- bool **showColumnSize**
- bool **showColumnFilename**
- detailsLayoutProperty **detailsLayout**

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

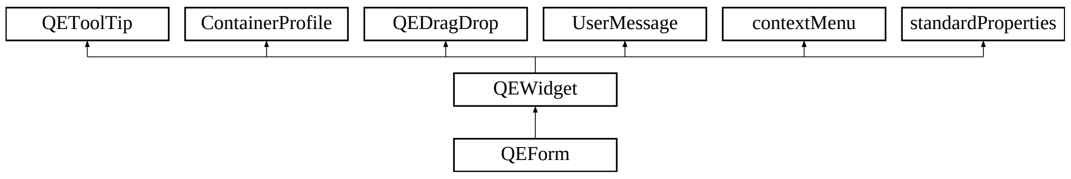
- double **formatFloating** (const QVariant &value)
- QVector< double > **formatFloatingArray** (const QVariant &value)
- QVariant **formatValue** (const double &floatingValue, generic::generic_types valueType)
- void **setPrecision** (unsigned int precision)
- void **setFormat** (formats format)
- unsigned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

9.60.1 Member Function Documentation

9.60.1.1 void QEForm::setVariableNameAndSubstitutions (QString *variableNameIn,* QString *variableNameSubstitutionsIn,* unsigned int *variableIndex*) [virtual]

Virtual function that may be implimented by users of QEWidget to update variable names and macro substitutions. A default is provided that is suitible in most cases.

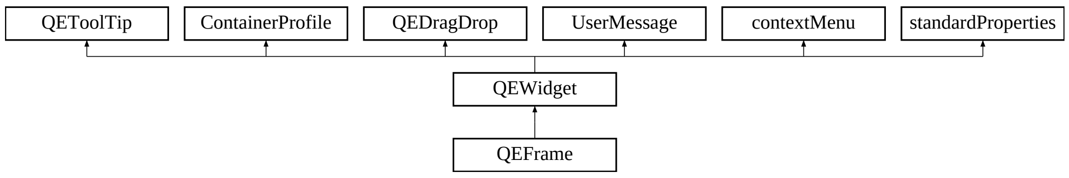
Reimplemented from QEWidget.

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 }

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 Enumeration Documentation

9.61.1.1 enum QEFrame::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.61.2 Member Function Documentation

9.61.2.1 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.

9.61.3 Property Documentation

9.61.3.1 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.

9.61.3.2 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.

9.61.3.3 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.

9.61.3.4 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 wide 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'.

9.61.3.5 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 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.

9.61.3.6 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.

9.61.3.7 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.

9.61.3.8 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 wide through the QELogin widget, or programatically through setUser-Level() 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'.

9.61.3.9 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 QEToolTip.

9.61.3.10 bool QEFrame::visible `[read, write]`

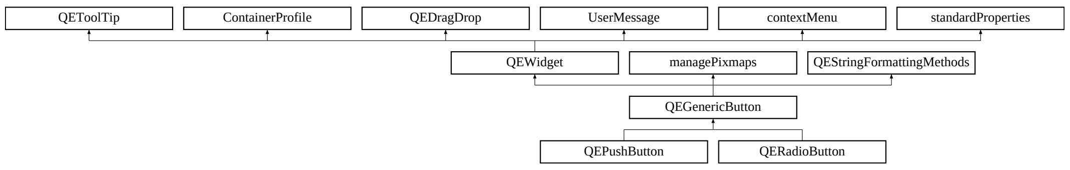
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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_A-ND_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** ()

Protected Member Functions

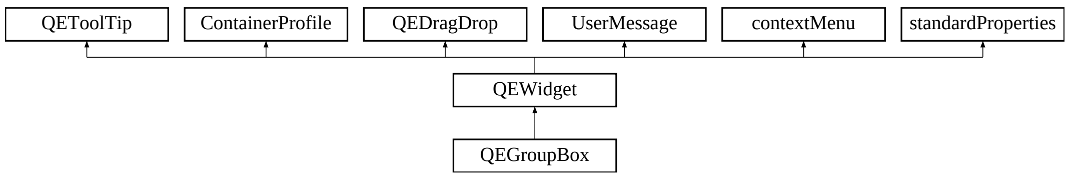
- void **connectionChanged** (QCaConnectionInfo &connectionInfo)
- void **setGenericButtonText** (const QString &text, QCaAlarmInfo &alarmInfo, Q-CaDateTime &, const unsigned int &variableIndex)
- void **userPressed** ()
- void **userReleased** ()
- void **userClicked** (bool checked)
- void **launchGui** (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 }

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 Enumeration Documentation

9.63.1.1 enum QEGroupBox::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.63.2 Member Function Documentation

9.63.2.1 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.

9.63.3 Property Documentation

9.63.3.1 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.

9.63.3.2 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.

9.63.3.3 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.

9.63.3.4 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 wide 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'.

9.63.3.5 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 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.

9.63.3.6 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 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.

9.63.3.7 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.

9.63.3.8 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 wide through the QELogin widget, or programatically through setUser-Level() 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'.

9.63.3.9 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 QEToolTip.

9.63.3.10 bool QEGroupBox::visible `[read, write]`

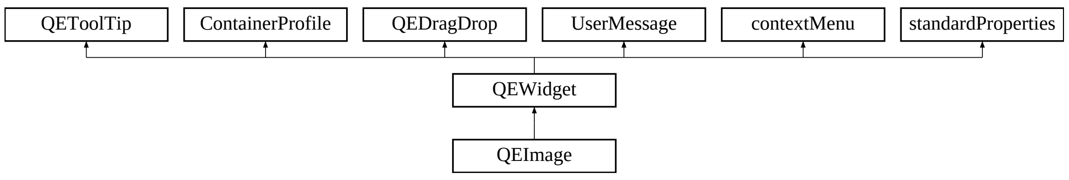
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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_SCIE-NTIST, Engineer = USERLEVEL_ENGINEER }
- enum **FormatOptions** { **Grey_8** = QEImage::GREY8, **Grey_12** = QEImage::GR-EY12, **Grey_16** = QEImage::GREY16, **RGB** = QEImage::RGB_888 }
- enum **ResizeOptions** { **Zoom** = QEImage::RESIZE_OPTION_ZOOM, **Fit** = QE-Image::RESIZE_OPTION_FIT }
- enum **RotationOptions** { **NoRotation** = QEImage::ROTATION_0, **Rotate90-Right** = QEImage::ROTATION_90_RIGHT, **Rotate90Left** = QEImage::ROTAT-ION_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**, **R-OI_H_VARIABLE**, **TARGET_X_VARIABLE**, **TARGET_Y_VARIABLE**, **BEAM-_X_VARIABLE**, **BEAM_Y_VARIABLE**, **TARGET_TRIGGER_VARIABLE**, **CL-IPPING_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 Enumeration Documentation

9.64.1.1 enum QEImage::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.64.2 Member Function Documentation

9.64.2.1 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.

9.64.2.2 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.

9.64.3 Property Documentation

9.64.3.1 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.

9.64.3.2 QString QEImage::beamXVariable `[read, write]`

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

9.64.3.3 QString QEImage::beamYVariable `[read, write]`

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

9.64.3.4 QString QEImage::clippingHighVariable `[read, write]`

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

9.64.3.5 QString QEImage::clippingLowVariable `[read, write]`

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

9.64.3.6 QString QEImage::clippingOnOffVariable `[read, write]`

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

9.64.3.7 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.

9.64.3.8 QString QEImage::heightVariable `[read, write]`

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

9.64.3.9 QString QEImage::imageVariable `[read, write]`

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

9.64.3.10 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.

9.64.3.11 QString QEImage::regionOfInterestHVariable `[read, write]`

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

9.64.3.12 QString QEImage::regionOfInterestWVariable `[read, write]`

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

9.64.3.13 QString QEImage::regionOfInterestXVariable `[read, write]`

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

9.64.3.14 QString QEImage::regionOfInterestYVariable `[read, write]`

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

9.64.3.15 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.

9.64.3.16 QString QEImage::targetXVariable [read, write]

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

9.64.3.17 QString QEImage::targetYVariable [read, write]

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

9.64.3.18 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 wide 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'.

9.64.3.19 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.

9.64.3.20 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.

9.64.3.21 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.

9.64.3.22 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 wide 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'.

9.64.3.23 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 QEToolTip.

9.64.3.24 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.

9.64.3.25 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.

9.64.3.26 QString QEImage::widthVariable [read, write]

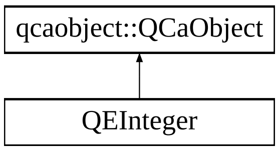
EPICS variable name (CA PV). This variable is used to read the width of the image.

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 value-Type)
- 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.

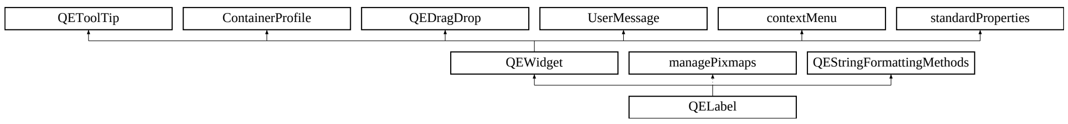
The documentation for this class was generated from the following files:

- /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 }
- 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 ArrayActions { Append = QEStringFormatting::APPEND, Ascii = QEStringFormatting::ASCII, Index = QEStringFormatting::INDEX }
- 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.

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::ArrayActions

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

Enumerator:

Append Refer to QEStringFormatting::APPEND for details.

Ascii Refer to QEStringFormatting::ASCII for details.

Index Refer to QEStringFormatting::INDEX for details.

9.67.2.2 enum QELabel::Formats

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

Enumerator:

Default Format as best appropriate for the data type.

Floating Format as a floating point number.

Integer Format as an integer.

UnsignedInteger Format as an unsigned integer.

Time Format as a time.

LocalEnumeration Format as a selection from the localEnumeration property.

9.67.2.3 enum QELabel::Notations

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

Enumerator:

Fixed Refer to QEStringFormatting::NOTATION_FIXED for details.

Scientific Refer to QEStringFormatting::NOTATION_SCIENTIFIC for details.

Automatic Refer to QEStringFormatting::NOTATION_AUTOMATIC for details.

9.67.2.4 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.

9.67.2.5 enum QELabel::UpdateOptions

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

Enumerator:

Text Data updates will update the label text.

Picture Data updates will update the label icon.

9.67.2.6 enum QELabel::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

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 E-PICS 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.

Format is:

[[<|<=|=|!=|>=|>]value1|∗] : string1 , [[<|<=|=|!=|>=|>]value2|∗] : string2 , [[<|<=|=|!=|>=|>]value3|∗] : string3 , ...

Where: < Less than <= Less than or equal = Equal (default if no operator specified) >= Greather than or equal > Greater than Always match (used to specify default text)

Values may be numeric or textual Values do not have to be in any order, but first match wins Values may be quoted Strings may be quoted Consecutive values do not have to be present. Operator is assumed to be equality if not present. White space is ignored except within quoted strings.

may be included in a string to indicate a line break

Examples are:

0:Off,1:On 0 : "Pump Running", 1 : "Pump not running" 0:"", 1:"Warning!\nAlarm" <2-:"Value is less than two", =2:"Value is equal to two", >2:"Value is grater than 2" 3:"-Beamline Available", ∗:"" "Pump Off":"OH NO!, the pump is OFF!","Pump On":"It's OK, the pump is on"

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:""'

A range of numbers can be covered by a pair of values as in the following example: >=4:"Between 4 and 8",<=8:"Between 4 and 8"

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 wide 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'.

9.67.5.23 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 wide through the QELogin widget, or programatically through setUser-Level() 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 QEToolTip.

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]`

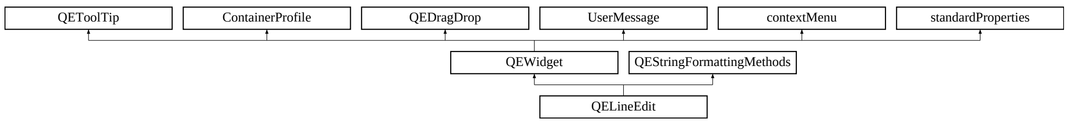
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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

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 Member Enumeration Documentation

9.68.1.1 enum QELineEdit::ArrayActions

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

Enumerator:

Append Refer to QEStringFormatting::APPEND for details.

Ascii Refer to QEStringFormatting::ASCII for details.

Index Refer to QEStringFormatting::INDEX for details.

9.68.1.2 enum QELineEdit::Formats

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

Enumerator:

Default Refer to QEStringFormatting::FORMAT_DEFAULT for details.

Floating Refer to QEStringFormatting::FORMAT_FLOATING for details.

Integer Refer to QEStringFormatting::FORMAT_INTEGER for details.

UnsignedInteger Refer to QEStringFormatting::FORMAT_UNSIGNEDINTEGE-R for details.

Time Refer to QEStringFormatting::FORMAT_TIME for details.

LocalEnumeration Refer to QEStringFormatting::FORMAT_LOCAL_ENUMER-ATE for details (and the localEnumeration property)

9.68.1.3 enum QELineEdit::Notations

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

Enumerator:

Fixed Refer to QEStringFormatting::NOTATION_FIXED for details.

Scientific Refer to QEStringFormatting::NOTATION_SCIENTIFIC for details.

Automatic Refer to QEStringFormatting::NOTATION_AUTOMATIC for details.

9.68.1.4 enum QELineEdit::UserLevels

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

Enumerator:

User Refer to ::USERLEVEL_USER for details.

Scientist Refer to ::USERLEVEL_SCIENTIST for details.

Engineer Refer to ::USERLEVEL_ENGINEER for details.

9.68.2 Constructor & Destructor Documentation

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

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

9.68.2.2 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.

9.68.3 Member Function Documentation

9.68.3.1 void QELineEdit::dbValueChanged (const QString & *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.

9.68.3.2 bool QELineEdit::getConfirmWrite ( )

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

9.68.3.3 bool QELineEdit::getSubscribe ( )

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

9.68.3.4 bool QELineEdit::getWriteOnEnter ( )

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

9.68.3.5 bool QELineEdit::getWriteOnFinish ( )

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

9.68.3.6 bool QELineEdit::getWriteOnLoseFocus ( )

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

9.68.3.7 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.

9.68.3.8 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).

9.68.3.9 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)

9.68.3.10 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').

9.68.3.11 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).

9.68.3.12 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).

9.68.4 Property Documentation

9.68.4.1 bool QELineEdit::addUnits `[read, write]`

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

9.68.4.2 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.

9.68.4.3 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'.

9.68.4.4 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).

9.68.4.5 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.

9.68.4.6 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).

9.68.4.7 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.

9.68.4.8 bool QELineEdit::leadingZero `[read, write]`

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

9.68.4.9 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.

Format is:

[[<|<=|=|!=|>=|>]value1|∗] : string1 , [[<|<=|=|!=|>=|>]value2|∗] : string2 , [[<|<=|=|!=|>=|>]value3|∗] : string3 , ...

Where: < Less than <= Less than or equal = Equal (default if no operator specified) >= Greather than or equal > Greater than Always match (used to specify default text)

Values may be numeric or textual Values do not have to be in any order, but first match wins Values may be quoted Strings may be quoted Consecutive values do not have to be present. Operator is assumed to be equality if not present. White space is ignored except within quoted strings.

may be included in a string to indicate a line break

Examples are:

0:Off,1:On 0 : "Pump Running", 1 : "Pump not running" 0:"", 1:"Warning!\nAlarm" <2-:"Value is less than two", =2:"Value is equal to two", >2:"Value is grater than 2" 3:"-Beamline Available", ∗:"" "Pump Off":"OH NO!, the pump is OFF!","Pump On":"It's OK, the pump is on"

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:""'

A range of numbers can be covered by a pair of values as in the following example: >=4:"Between 4 and 8",<=8:"Between 4 and 8"

9.68.4.10 Notations QELineEdit::notation `[read, write]`

Notation used for numerical formatting. Default is fixed.

9.68.4.11 int QELineEdit::precision `[read, write]`

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

9.68.4.12 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.

9.68.4.13 bool QELineEdit::trailingZeros [read, write]

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

9.68.4.14 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.

9.68.4.15 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 wide 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'.

9.68.4.16 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 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.

9.68.4.17 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.

9.68.4.18 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.

9.68.4.19 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 wide through the QELogin widget, or programatically through setUser-Level() 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'.

9.68.4.20 QString QELineEdit::variable `[read, write]`

EPICS variable name (CA PV)

9.68.4.21 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 QEToolTip.

9.68.4.22 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.

9.68.4.23 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.

9.68.4.24 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').

9.68.4.25 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).

9.68.4.26 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

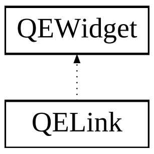
- **QELineEditManager** (QObject ∗parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget ∗ **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** = QE-Link::CONDITION_NE, **GreaterThan** = QELink::CONDITION_GT, **GreaterThan-OrEqual** = 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

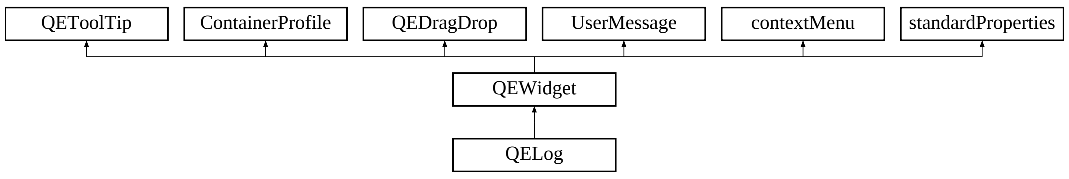
- ConditionNames **condition**
- QString **comparisonValue**
- QString **outTrueValue**
- QString **outFalseValue**
- bool **runVisible**

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

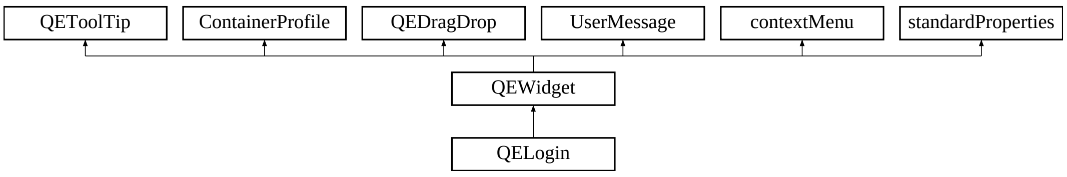
- bool **showColumnTime**
- bool **showColumnType**
- bool **showColumnMessage**
- bool **showMessageFilter**
- bool **showClear**
- bool **showSave**
- detailsLayoutProperty **detailsLayout**
- QColor **infoColor**
- QColor **warningColor**
- QColor **errorColor**
- MessageFilterOptions **messageFormFilter**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

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

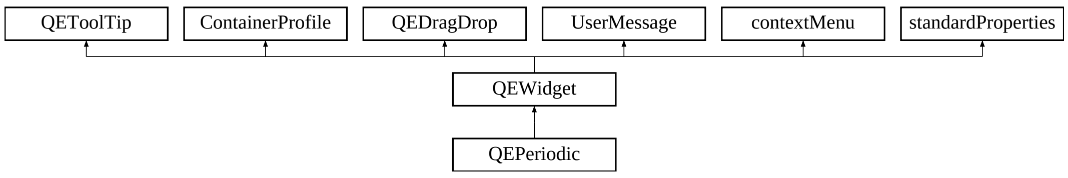
- bool **showUserType**
- bool **showLogin**
- bool **showLogout**
- userTypesProperty **currentUserType**
- detailsLayoutProperty **detailsLayout**

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 }
- 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 Enumeration Documentation

9.73.1.1 enum QEPeriodic::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.73.2 Member Function Documentation

9.73.2.1 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.

9.73.2.2 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.

9.73.2.3 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.

9.73.3 Member Data Documentation

9.73.3.1 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.

9.73.4 Property Documentation

9.73.4.1 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.

9.73.4.2 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.

9.73.4.3 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.

9.73.4.4 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.

9.73.4.5 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.

9.73.4.6 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 wide 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'.

9.73.4.7 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 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.

9.73.4.8 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 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.

9.73.4.9 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.

9.73.4.10 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 wide through the QELogin widget, or programatically through setUser-Level() 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'.

9.73.4.11 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 QEToolTip.

9.73.4.12 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.

9.73.4.13 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.

9.73.4.14 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.

9.73.4.15 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

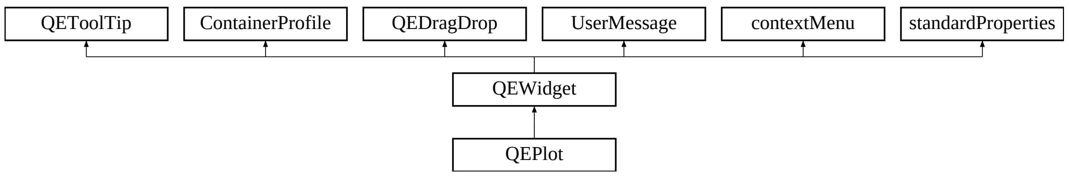
- static void **setDebugLevel** (unsigned level=0)
- static QVariant **get** (const QString &_pvName, int delay=defaultDelay)
- static QVariant **set** (QString &_pvName, const QVariant &value, int delay=-1)

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 }
- 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 Enumeration Documentation

9.78.1.1 enum QEPlot::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.78.2 Member Function Documentation

9.78.2.1 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.

9.78.2.2 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.

9.78.2.3 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.

9.78.3 Member Data Documentation

9.78.3.1 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.

9.78.4 Property Documentation

9.78.4.1 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.

9.78.4.2 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.

9.78.4.3 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 wide 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'.

9.78.4.4 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 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.

9.78.4.5 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 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.

9.78.4.6 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.

9.78.4.7 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 wide through the QELogin widget, or programatically through setUser-Level() 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'.

9.78.4.8 QString QEPlot::variable1 [read, write]

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

9.78.4.9 QString QEPlot::variable2 [read, write]

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

9.78.4.10 QString QEPlot::variable3 [read, write]

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

9.78.4.11 QString QEPlot::variable4 [read, write]

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

9.78.4.12 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 QEToolTip.

9.78.4.13 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.

9.78.4.14 bool QEPlot::visible `[read, write]`

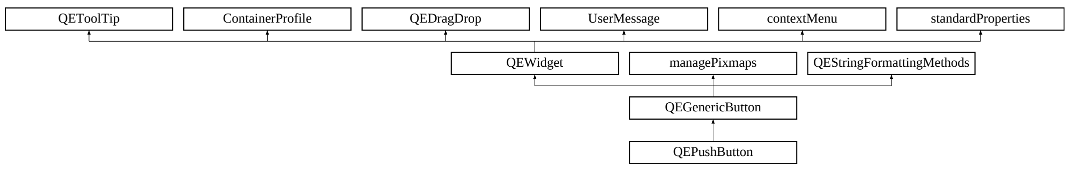
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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_SCIE-NTIST, Engineer = USERLEVEL_ENGINEER }
- enum Formats { Default = QEStringFormatting::FORMAT_DEFAULT, Floating = QEStringFormatting::FORMAT_FLOATING, Integer = QEStringFormatting::FO-RMAT_INTEGER, UnsignedInteger = QEStringFormatting::FORMAT_UNSIGN-EDINTEGER, Time = QEStringFormatting::FORMAT_TIME, LocalEnumeration = QEStringFormatting::FORMAT_LOCAL_ENUMERATE }
- enum Notations { Fixed = QEStringFormatting::NOTATION_FIXED, Scientific = QEStringFormatting::NOTATION_SCIENTIFIC, Automatic = QEString-Formatting::NOTATION_AUTOMATIC }
- enum ArrayActions { Append = QEStringFormatting::APPEND, Ascii = QEString-Formatting::ASCII, Index = QEStringFormatting::INDEX }
- enum UpdateOptions { Text = QEGenericButton::UPDATE_TEXT, Icon = QE-GenericButton::UPDATE_ICON, TextAndIcon = QEGenericButton::UPDATE_T-EXT_AND_ICON, State = QEGenericButton::UPDATE_STATE }

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

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

  Creation options. Used to indicate how to present a GUI when requesting a new GUI be created. Open a new window, open a new tab, or replace the current window.

Public Slots

- void launchGui (QString guiName, QEForm::creationOptions creationOption)
- 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.
- 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.

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
- int precision
- bool useDbPrecision
- bool leadingZero
- bool trailingZeros
- bool addUnits
- QString localEnumeration
- Formats format
- Notations notation
- ArrayActions arrayAction
- Qt::Alignment alignment
- UpdateOptions updateOption
- QPixmap pixmap0
- QPixmap pixmap1
- QPixmap pixmap2
- QPixmap pixmap3
- QPixmap pixmap4
- QPixmap pixmap5
- QPixmap pixmap6

- QPixmap pixmap7
- 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 Enumeration Documentation

9.79.1.1 enum QEPushButton::ArrayActions

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

Enumerator:

Append Refer to QEStringFormatting::APPEND for details.

Ascii Refer to QEStringFormatting::ASCII for details.

Index Refer to QEStringFormatting::INDEX for details.

9.79.1.2 enum QEPushButton::CreationOptionNames

Creation options. Used to indicate how to present a GUI when requesting a new GUI be created. Open a new window, open a new tab, or replace the current window.

Enumerator:

Open Replace the current GUI with the new GUI.

NewTab Open new GUI in a new tab.

NewWindow Open new GUI in a new window.

9.79.1.3 enum QEPushButton::Formats

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

Enumerator:

Default Format as best appropriate for the data type.

Floating Format as a floating point number.

Integer Format as an integer.

UnsignedInteger Format as an unsigned integer.

Time Format as a time.

LocalEnumeration Format as a selection from the localEnumeration property.

9.79.1.4 enum QEPushButton::Notations

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

Enumerator:

Fixed Refer to QEStringFormatting::NOTATION_FIXED for details.

Scientific Refer to QEStringFormatting::NOTATION_SCIENTIFIC for details.

Automatic Refer to QEStringFormatting::NOTATION_AUTOMATIC for details.

9.79.1.5 enum QEPushButton::UpdateOptions

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

Enumerator:

Text Data updates will update the button text.

Icon Data updates will update the button icon.

TextAndIcon Data updates will update the button text and icon.

State Data updates will update the button state (checked or unchecked)

9.79.1.6 enum QEPushButton::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.79.2 Constructor & Destructor Documentation

9.79.2.1 QEPushButton::QEPushButton (QWidget ∗ *parent* = 0)

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

9.79.2.2 QEPushButton::QEPushButton (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.

9.79.3 Member Function Documentation

9.79.3.1 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.

9.79.3.2 void QEPushButton::launchGui (QString *guiName,* QEForm::creationOptions *creationOption*) `[inline, slot]`

Default slot used to create a new GUI if there is no slot indicated in the ContainerProfile class. This slot is typically used when the button is pressed within the Designer preview window to allow the operation of the button to be tested. If an application does not specify a slot to use for creating new windows (through the ContainerProfile class) a window will still be created through this slot, but it will not respect the window creation options or any other window related application constraints. For example, the QEGui application does provide a slot for creating new GUIs in the ContainerProfile class which respects the creation options, knows how to add tabs in the application, and extend the application's window menu in the menu bar.

Reimplemented from QEGenericButton.

9.79.3.3 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.

9.79.4 Property Documentation

9.79.4.1 bool QEPushButton::addUnits `[read, write]`

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

9.79.4.2 Qt::Alignment QEPushButton::alignment `[read, write]`

Set the buttons text alignment. Left justification is particularly useful when displaying quickly changing numeric data updates.

9.79.4.3 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.

9.79.4.4 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.

9.79.4.5 QStringList QEPushButton::arguments `[read, write]`

Arguments for program specified in the 'program' property.

Reimplemented from QEGenericButton.

9.79.4.6 ArrayActions QEPushButton::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'.

9.79.4.7 QString QEPushButton::clickCheckedText [read, write]

Text used to compare with text written or read to determine if push button should be marked as checked. Note, must be an exact match following formatting of data updates. When writing values, the 'pressText', 'ReleaseText', or 'clickedtext' must match this property to cause the button to be checked when the write occurs.

Good example: formatting set to diaplay a data value of '1' as 'On', clickCheckedText is 'On', clickText is 'On'. In this example, the push button will be checked when a data update occurs with a value of 1 or when the button is clicked.

Bad example: formatting set to diaplay a data value of '1' as 'On', clickCheckedText is 'On', clickText is '1'. In this example, the push button will be checked when a data update occurs with a value of 1 but, although a valid value will be written when clicked, the button will not be checked when clicked as '1' is not the same as 'On'.

Reimplemented from QEGenericButton.

9.79.4.8 QString QEPushButton::clickText [read, write]

Value written when user clicks button if 'writeOnClick' property is true

Reimplemented from QEGenericButton.

9.79.4.9 bool QEPushButton::confirmAction [read, write]

If true, a dialog will be presented asking the user to confirm if the button action should be carried out

9.79.4.10 CreationOptionNames QEPushButton::creationOption [read, write]

Creation options when opening a new GUI. Open a new window, open a new tab, or replace the current window. the creation option is supplied when the button generates a newGui signal. Application code connected to this signal should honour this request if possible. When used within the QEGui application, the QEGui application creates a new window, new tab, or replaces the current window as appropriate.

Reimplemented from QEGenericButton.

9.79.4.11 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.

9.79.4.12 Formats QEPushButton::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).

9.79.4.13 QString QEPushButton::guiFile `[read, write]`

File name of GUI to be presented on button click. File name can be absolute, relative to the path of the QEform in which the QEPushButton is located, relative to the any path in the path list published in the ContainerProfile class, or relative to the current path. See QEWidget::openQEFile() in QEWidget.cpp for details.

9.79.4.14 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.

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.

9.79.4.15 QString QEPushButton::labelText `[read, write]`

Button label text (prior to substitution). Macro substitutions will be applied to this text and the result will be set as the button text. Used when data updates are not being represented in the button text. For example, a button in a sub form may have a 'labelText' property of 'Turn Pump On'. When the sub form is used twice in a main form with substitutions PUMPNUM=1 and PUMPNUM=2 respectively, the two identical buttons in the sub forms will have the labels 'Turn Pump 1 On' and 'Turn Pump 2 On' respectively.

Reimplemented from QEGenericButton.

9.79.4.16 bool QEPushButton::leadingZero `[read, write]`

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

9.79.4.17 QString QEPushButton::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.

Format is:

[[<|<=|=|!=|>=|>]value1|∗] : string1 , [[<|<=|=|!=|>=|>]value2|∗] : string2 , [[<|<=|=|!=|>=|>]value3|∗] : string3 , ...

Where: < Less than <= Less than or equal = Equal (default if no operator specified) >= Greather than or equal > Greater than Always match (used to specify default text)

Values may be numeric or textual Values do not have to be in any order, but first match wins Values may be quoted Strings may be quoted Consecutive values do not have to be present. Operator is assumed to be equality if not present. White space is ignored except within quoted strings.

may be included in a string to indicate a line break

Examples are:

0:Off,1:On 0 : "Pump Running", 1 : "Pump not running" 0:"", 1:"Warning!\nAlarm" <2-:"Value is less than two", =2:"Value is equal to two", >2:"Value is grater than 2" 3:"-Beamline Available", ∗:"" "Pump Off":"OH NO!, the pump is OFF!","Pump On":"It's OK, the pump is on"

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:""'

A range of numbers can be covered by a pair of values as in the following example: >=4:"Between 4 and 8",<=8:"Between 4 and 8"

9.79.4.18 Notations QEPushButton::notation `[read, write]`

Notation used for numerical formatting. Default is fixed.

9.79.4.19 QString QEPushButton::password `[read, write]`

Password user will need to enter before any action is taken

Reimplemented from QEGenericButton.

9.79.4.20 QPixmap QEPushButton::pixmap0 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 0

9.79.4.21 QPixmap QEPushButton::pixmap1 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 1

9.79.4.22 QPixmap QEPushButton::pixmap2 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 2

9.79.4.23 QPixmap QEPushButton::pixmap3 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 3

9.79.4.24 QPixmap QEPushButton::pixmap4 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 4

9.79.4.25 QPixmap QEPushButton::pixmap5 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 5

9.79.4.26 QPixmap QEPushButton::pixmap6 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 6

9.79.4.27 QPixmap QEPushButton::pixmap7 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 7

9.79.4.28 int QEPushButton::precision `[read, write]`

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

9.79.4.29 QString QEPushButton::pressText `[read, write]`

Value written when user presses button if 'writeOnPress' property is true

Reimplemented from QEGenericButton.

9.79.4.30 QString QEPushButton::program [read, write]

Program to run when the button is clicked. No attempt to run a program is made if this property is empty. Example: firefox

Reimplemented from QEGenericButton.

9.79.4.31 QString QEPushButton::releaseText [read, write]

Value written when user releases button if 'writeOnRelease' property is true

Reimplemented from QEGenericButton.

9.79.4.32 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.

9.79.4.33 bool QEPushButton::trailingZeros [read, write]

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

9.79.4.34 UpdateOptions QEPushButton::updateOption [read, write]

Update options (text, pixmap, both, or state (checked or unchecked)

Reimplemented from QEGenericButton.

9.79.4.35 bool QEPushButton::useDbPrecision [read, write]

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

9.79.4.36 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 wide 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'.

9.79.4.37 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 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.

9.79.4.38 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 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.

9.79.4.39 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.

9.79.4.40 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 wide through the QELogin widget, or programatically through setUser- Level() 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'.

9.79.4.41 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.

9.79.4.42 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 QEToolTip.

9.79.4.43 QString QEPushButton::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.

9.79.4.44 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.

9.79.4.45 bool QEPushButton::writeOnClick `[read, write]`

If true, the 'clickText' property is written when the button is clicked. Default is true

Reimplemented from QEGenericButton.

9.79.4.46 bool QEPushButton::writeOnPress `[read, write]`

If true, the 'pressText' property is written when the button is pressed. Default is false

Reimplemented from QEGenericButton.

9.79.4.47 bool QEPushButton::writeOnRelease `[read, write]`

If true, the 'releaseText' property is written when the button is released. Default is false

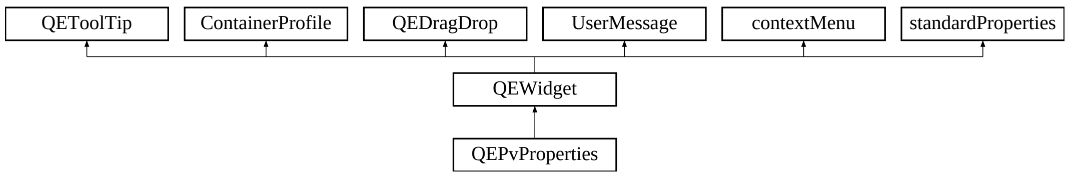
Reimplemented from QEGenericButton.

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_SCIE-NTIST, Engineer = USERLEVEL_ENGINEER }

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)

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 Enumeration Documentation

9.80.1.1 enum QEPvProperties::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.80.2 Member Function Documentation

9.80.2.1 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.

9.80.3 Property Documentation

9.80.3.1 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.

9.80.3.2 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.

9.80.3.3 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.

9.80.3.4 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 wide 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'.

9.80.3.5 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 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.

9.80.3.6 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.

9.80.3.7 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.

9.80.3.8 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 wide through the QELogin widget, or programatically through setUser-Level() 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'.

9.80.3.9 QString QEPvProperties::variable `[read, write]`

EPICS variable name (CA PV)

9.80.3.10 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 QEToolTip.

9.80.3.11 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.

9.80.3.12 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

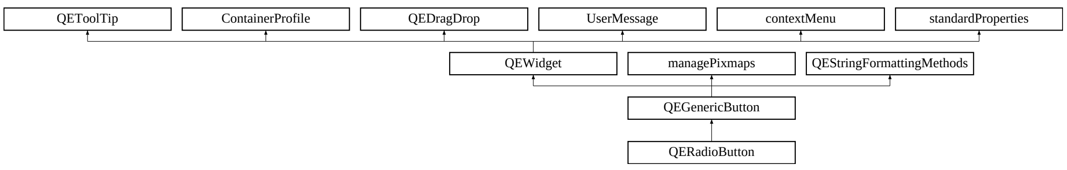
- **QEPvPropertiesManager** (QObject ∗parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget ∗ **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_SCIE-NTIST, Engineer = USERLEVEL_ENGINEER }
- enum Formats { Default = QEStringFormatting::FORMAT_DEFAULT, Floating = QEStringFormatting::FORMAT_FLOATING, Integer = QEStringFormatting::FO-RMAT_INTEGER, UnsignedInteger = QEStringFormatting::FORMAT_UNSIGN-EDINTEGER, Time = QEStringFormatting::FORMAT_TIME, LocalEnumeration = QEStringFormatting::FORMAT_LOCAL_ENUMERATE }
- enum Notations { Fixed = QEStringFormatting::NOTATION_FIXED, Scientific = QEStringFormatting::NOTATION_SCIENTIFIC, Automatic = QEString-Formatting::NOTATION_AUTOMATIC }
- enum ArrayActions { Append = QEStringFormatting::APPEND, Ascii = QEString-Formatting::ASCII, Index = QEStringFormatting::INDEX }
- enum UpdateOptions { Text = QEGenericButton::UPDATE_TEXT, Icon = QE-GenericButton::UPDATE_ICON, TextAndIcon = QEGenericButton::UPDATE_T-EXT_AND_ICON, State = QEGenericButton::UPDATE_STATE }

  User friendly enumerations for updateOption property - refer to QEGenericButton-::updateOptions for details.
- enum CreationOptionNames { Open = QEForm::CREATION_OPTION_OPEN, -NewTab = QEForm::CREATION_OPTION_NEW_TAB, NewWindow = QEForm:-:CREATION_OPTION_NEW_WINDOW }

  Creation options. Used to indicate how to present a GUI when requesting a new GUI be created. Open a new window, open a new tab, or replace the current window.

Public Slots

- void launchGui (QString guiName, QEForm::creationOptions creationOption)
- 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.
- 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.

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
- int precision
- bool useDbPrecision
- bool leadingZero
- bool trailingZeros
- bool addUnits
- QString localEnumeration
- Formats format
- Notations notation
- ArrayActions arrayAction
- Qt::Alignment alignment
- UpdateOptions updateOption
- QPixmap pixmap0
- QPixmap pixmap1
- QPixmap pixmap2
- QPixmap pixmap3
- QPixmap pixmap4
- QPixmap pixmap5
- QPixmap pixmap6
- QPixmap pixmap7
- 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 Enumeration Documentation

9.82.1.1 enum QERadioButton::ArrayActions

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

Enumerator:

Append Refer to QEStringFormatting::APPEND for details.

Ascii Refer to QEStringFormatting::ASCII for details.

Index Refer to QEStringFormatting::INDEX for details.

9.82.1.2 enum QERadioButton::CreationOptionNames

Creation options. Used to indicate how to present a GUI when requesting a new GUI be created. Open a new window, open a new tab, or replace the current window.

Enumerator:

Open Replace the current GUI with the new GUI.

NewTab Open new GUI in a new tab.

NewWindow Open new GUI in a new window.

9.82.1.3 enum QERadioButton::Formats

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

Enumerator:

Default Format as best appropriate for the data type.

Floating Format as a floating point number.

Integer Format as an integer.

UnsignedInteger Format as an unsigned integer.

Time Format as a time.

LocalEnumeration Format as a selection from the localEnumeration property.

9.82.1.4 enum QERadioButton::Notations

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

Enumerator:

Fixed Refer to QEStringFormatting::NOTATION_FIXED for details.

Scientific Refer to QEStringFormatting::NOTATION_SCIENTIFIC for details.

Automatic Refer to QEStringFormatting::NOTATION_AUTOMATIC for details.

9.82.1.5 enum QERadioButton::UpdateOptions

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

Enumerator:

Text Data updates will update the button text.

Icon Data updates will update the button icon.

TextAndIcon Data updates will update the button text and icon.

State Data updates will update the button state (checked or unchecked)

9.82.1.6 enum QERadioButton::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.82.2 Constructor & Destructor Documentation

9.82.2.1 QERadioButton::QERadioButton (QWidget * *parent* = 0)

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

9.82.2.2 QERadioButton::QERadioButton (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.

9.82.3 Member Function Documentation

9.82.3.1 void QERadioButton::dbValueChanged (const QString & *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.

9.82.3.2 void QERadioButton::launchGui (QString *guiName*, QEForm::creationOptions *creationOption*) [inline, slot]

Default slot used to create a new GUI if there is no slot indicated in the ContainerProfile class. This slot is typically used when the button is pressed within the Designer preview window to allow the operation of the button to be tested. If an application does not specify a slot to use for creating new windows (through the ContainerProfile class) a window will still be created through this slot, but it will not respect the window creation options or any other window related application constraints. For example, the QEGui application does provide a slot for creating new GUIs in the ContainerProfile class which respects the creation options, knows how to add tabs in the application, and extend the application's window menu in the menu bar.

Reimplemented from QEGenericButton.

9.82.3.3 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.

9.82.4 Property Documentation

9.82.4.1 bool QERadioButton::addUnits [read, write]

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

9.82.4.2 Qt::Alignment QERadioButton::alignment [read, write]

Set the buttons text alignment. Left justification is particularly useful when displaying quickly changing numeric data updates.

9.82.4.3 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.

9.82.4.4 QStringList QERadioButton::arguments [read, write]

Arguments for program specified in the 'program' property.

Reimplemented from QEGenericButton.

9.82.4.5 ArrayActions QERadioButton::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'.

9.82.4.6 QString QERadioButton::clickCheckedText [read, write]

Text used to compare with text written or read to determine if push button should be marked as checked. Note, must be an exact match following formatting of data updates. When writing values, the 'pressText', 'ReleaseText', or 'clickedtext' must match this property to cause the button to be checked when the write occurs.

Good example: formatting set to diaplay a data value of '1' as 'On', clickCheckedText is 'On', clickText is 'On'. In this example, the push button will be checked when a data update occurs with a value of 1 or when the button is clicked.

Bad example: formatting set to diaplay a data value of '1' as 'On', clickCheckedText is 'On', clickText is '1'. In this example, the push button will be checked when a data update occurs with a value of 1 but, although a valid value will be written when clicked, the button will not be checked when clicked as '1' is not the same as 'On'.

Reimplemented from QEGenericButton.

9.82.4.7 QString QERadioButton::clickText [read, write]

Value written when user clicks button if 'writeOnClick' property is true

Reimplemented from QEGenericButton.

9.82.4.8 bool QERadioButton::confirmAction [read, write]

If true, a dialog will be presented asking the user to confirm if the button action should be carried out

9.82.4.9 CreationOptionNames QERadioButton::creationOption [read, write]

Creation options when opening a new GUI. Open a new window, open a new tab, or replace the current window. the creation option is supplied when the button generates a newGui signal. Application code connected to this signal should honour this request if possible. When used within the QEGui application, the QEGui application creates a new window, new tab, or replaces the current window as appropriate.

Reimplemented from QEGenericButton.

9.82.4.10 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.

9.82.4.11 Formats QERadioButton::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).

9.82.4.12 QString QERadioButton::guiFile `[read, write]`

File name of GUI to be presented on button click. File name can be absolute, relative to the path of the QEform in which the QEPushButton is located, relative to the any path in the path list published in the ContainerProfile class, or relative to the current path. See QEWidget::openQEFile() in QEWidget.cpp for details.

9.82.4.13 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.

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.

9.82.4.14 QString QERadioButton::labelText `[read, write]`

Button label text (prior to substitution). Macro substitutions will be applied to this text and the result will be set as the button text. Used when data updates are not being represented in the button text. For example, a button in a sub form may have a 'labelText' property of 'Turn Pump On'. When the sub form is used twice in a main form with substitutions PUMPNUM=1 and PUMPNUM=2 respectively, the two identical buttons in the sub forms will have the labels 'Turn Pump 1 On' and 'Turn Pump 2 On' respectively.

Reimplemented from QEGenericButton.

9.82.4.15 bool QERadioButton::leadingZero `[read, write]`

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

9.82.4.16 QString QERadioButton::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.

Format is:

[[<|<=|=|!=|>=|>]value1|∗] : string1 , [[<|<=|=|!=|>=|>]value2|∗] : string2 , [[<|<=|=|!=|>=|>]value3|∗] : string3 , ...

Where: < Less than <= Less than or equal = Equal (default if no operator specified) >= Greather than or equal > Greater than Always match (used to specify default text)

Values may be numeric or textual Values do not have to be in any order, but first match wins Values may be quoted Strings may be quoted Consecutive values do not have to be present. Operator is assumed to be equality if not present. White space is ignored except within quoted strings.

may be included in a string to indicate a line break

Examples are:

0:Off,1:On 0 : "Pump Running", 1 : "Pump not running" 0:"", 1:"Warning!\nAlarm" <2-:"Value is less than two", =2:"Value is equal to two", >2:"Value is grater than 2" 3:"-Beamline Available", ∗:"" "Pump Off":"OH NO!, the pump is OFF!","Pump On":"It's OK, the pump is on"

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:""'

A range of numbers can be covered by a pair of values as in the following example: >=4:"Between 4 and 8",<=8:"Between 4 and 8"

9.82.4.17 Notations QERadioButton::notation `[read, write]`

Notation used for numerical formatting. Default is fixed.

9.82.4.18 QString QERadioButton::password `[read, write]`

Password user will need to enter before any action is taken

Reimplemented from QEGenericButton.

9.82.4.19 QPixmap QERadioButton::pixmap0 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 0

9.82.4.20 QPixmap QERadioButton::pixmap1 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 1

9.82.4.21 QPixmap QERadioButton::pixmap2 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 2

9.82.4.22 QPixmap QERadioButton::pixmap3 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 3

9.82.4.23 QPixmap QERadioButton::pixmap4 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 4

9.82.4.24 QPixmap QERadioButton::pixmap5 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 5

9.82.4.25 QPixmap QERadioButton::pixmap6 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 6

9.82.4.26 QPixmap QERadioButton::pixmap7 `[read, write]`

Pixmap to display if updateOption is Icon or TextAndIcon and data value translates to an index of 7

9.82.4.27 int QERadioButton::precision `[read, write]`

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

9.82.4.28 QString QERadioButton::pressText `[read, write]`

Value written when user presses button if 'writeOnPress' property is true

Reimplemented from QEGenericButton.

9.82.4.29 QString QERadioButton::program `[read, write]`

Program to run when the button is clicked. No attempt to run a program is made if this property is empty. Example: firefox

Reimplemented from QEGenericButton.

9.82.4.30 QString QERadioButton::releaseText `[read, write]`

Value written when user releases button if 'writeOnRelease' property is true

Reimplemented from QEGenericButton.

9.82.4.31 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.

9.82.4.32 bool QERadioButton::trailingZeros `[read, write]`

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

9.82.4.33 UpdateOptions QERadioButton::updateOption `[read, write]`

Update options (text, pixmap, both, or state (checked or unchecked)

Reimplemented from QEGenericButton.

9.82.4.34 bool QERadioButton::useDbPrecision `[read, write]`

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

9.82.4.35 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 wide 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'.

9.82.4.36 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 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.

9.82.4.37 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 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.

9.82.4.38 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.

9.82.4.39 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 wide through the QELogin widget, or programatically through setUser-Level() 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'.

9.82.4.40 QString QERadioButton::variable `[read, write]`

EPICS variable name (CA PV)

9.82.4.41 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 QEToolTip.

9.82.4.42 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.

9.82.4.43 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.

9.82.4.44 bool QERadioButton::writeOnClick [read, write]

If true, the 'clickText' property is written when the button is clicked. Default is true

Reimplemented from QEGenericButton.

9.82.4.45 bool QERadioButton::writeOnPress [read, write]

If true, the 'pressText' property is written when the button is pressed. Default is false

Reimplemented from QEGenericButton.

9.82.4.46 bool QERadioButton::writeOnRelease [read, write]

If true, the 'releaseText' property is written when the button is released. Default is false

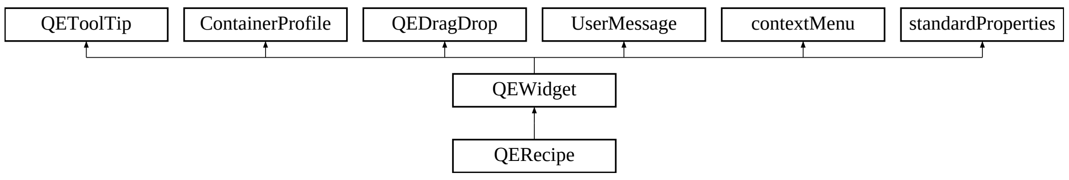
Reimplemented from QEGenericButton.

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 pConfigurationType)

- 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

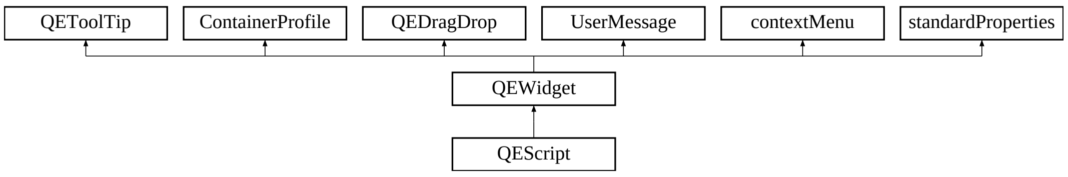
- QString **recipeDescription**
- bool **showRecipeList**
- bool **showNew**
- bool **showSave**
- bool **showDelete**
- bool **showApply**
- bool **showRead**
- bool **showFields**
- configurationTypesProperty **configurationType**
- QString **configurationFile**
- QString **configurationText**
- detailsLayoutProperty **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**

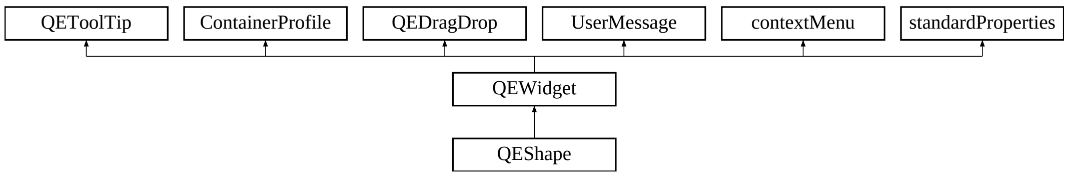
The documentation for this class was generated from the following files:

- /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_SCIE-NTIST, Engineer = USERLEVEL_ENGINEER }

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.

9.85.2.3 enum QEShape::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

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 wide 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'.

9.85.5.42 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 wide through the QELogin widget, or programatically through setUser-Level() 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 QEToolTip.

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]`

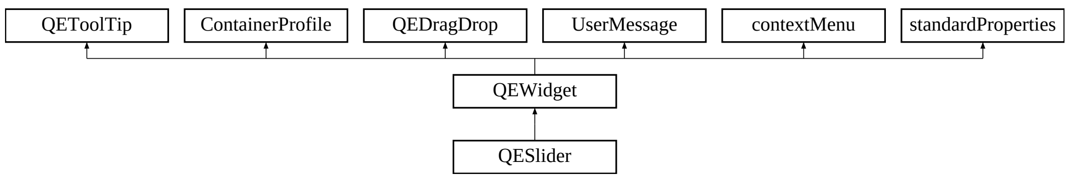
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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_SCIE-NTIST, Engineer = USERLEVEL_ENGINEER }

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 Enumeration Documentation

9.86.1.1 enum QESlider::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.86.2 Member Function Documentation

9.86.2.1 void QESlider::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.

9.86.2.2 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.

9.86.3 Member Data Documentation

9.86.3.1 bool QESlider::writeOnChange `[read, write, protected]`

Sets if this widget writes any changes as the user moves the slider (the QSlider 'value-Changed' signal is emitted). Default is 'true' (writes any changes when the QSlider 'valueChanged' signal is emitted).

9.86.4 Property Documentation

9.86.4.1 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.

9.86.4.2 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.

9.86.4.3 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.

9.86.4.4 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.

9.86.4.5 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 wide 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'.

9.86.4.6 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 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.

9.86.4.7 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 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.

9.86.4.8 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.

9.86.4.9 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 wide through the QELogin widget, or programatically through setUser-Level() 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'.

9.86.4.10 QString QESlider::variable `[read, write]`

EPICS variable name (CA PV)

9.86.4.11 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 QEToolTip.

9.86.4.12 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.

9.86.4.13 bool QESlider::visible `[read, write]`

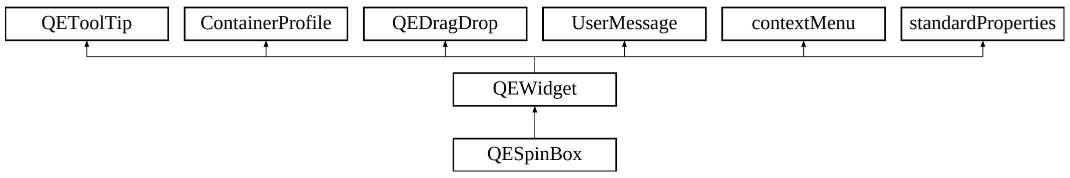
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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_SCIE-NTIST, Engineer = USERLEVEL_ENGINEER }

Public Slots

- void requestEnabled (const bool &state)

Signals

- void dbValueChanged (const double &out)
- void userChange (const QString &oldValue, const QString &newValue, const Q-String &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 Enumeration Documentation

9.87.1.1 enum QESpinBox::UserLevels

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

Enumerator:

User Refer to USERLEVEL_USER for details.

Scientist Refer to USERLEVEL_SCIENTIST for details.

Engineer Refer to USERLEVEL_ENGINEER for details.

9.87.2 Member Function Documentation

9.87.2.1 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.

9.87.2.2 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.

9.87.3 Property Documentation

9.87.3.1 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.

9.87.3.2 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.

9.87.3.3 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.

9.87.3.4 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.

9.87.3.5 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 wide 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'.

9.87.3.6 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.

9.87.3.7 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.

9.87.3.8 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.

9.87.3.9 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 wide through the QELogin widget, or programatically through setUser-Level() 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'.

9.87.3.10 QString QESpinBox::variable `[read, write]`

EPICS variable name (CA PV)

9.87.3.11 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 QEToolTip.

9.87.3.12 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.

9.87.3.13 bool QESpinBox::visible `[read, write]`

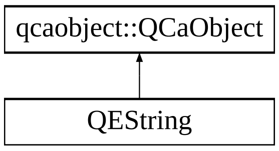
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a QELink widget. Note, when false the widget will still be visible in Qt Designer.

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, QCa-DateTime &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 ∗userMessage-In)
- 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_INTEG-ER, FORMAT_UNSIGNEDINTEGER, FORMAT_TIME, FORMAT_LOCAL_EN-UMERATE, FORMAT_STRING }

- enum notations { NOTATION_FIXED = QTextStream::FixedNotation, NOTATIO-N_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** ()

9.89.1 Member Enumeration Documentation

9.89.1.1 enum QEStringFormatting::arrayActions

What action to take when formatting array data

Enumerator:

APPEND Interpret each element in the array as an unsigned integer and append string representations of each element from the array with a space in between each.

ASCII Interpret each element from the array as a character in a string. Translate all non printing characters to '?' except for trailing zeros (ignore them)

INDEX Interpret the element selected by setArrayIndex() as an unsigned integer.

9.89.1.2 enum QEStringFormatting::formats

Formatting options

Enumerator:

FORMAT_DEFAULT Format according to the EPICS database record type.

FORMAT_FLOATING Format as a floating point number.

FORMAT_INTEGER Format as an integer.

FORMAT_UNSIGNEDINTEGER Format as an unsigned integer.

FORMAT_TIME Format as a time.

FORMAT_LOCAL_ENUMERATE Format as a selection from the local enumerations set by setLocalEnumeration()

FORMAT_STRING Format as a string.

9.89.1.3 enum QEStringFormatting::notations

Notations when formatting a floating point number

Enumerator:

NOTATION_FIXED Standard floating point 123456.789.

NOTATION_SCIENTIFIC Scientific representation 1.23456789e6.

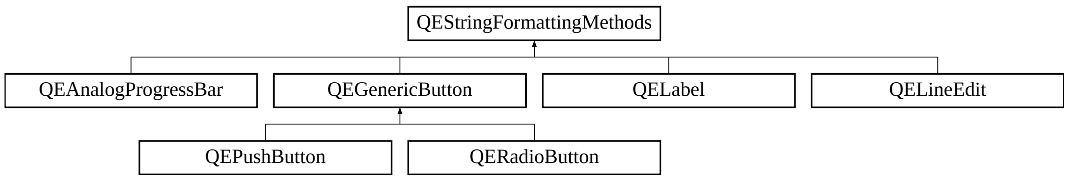
NOTATION_AUTOMATIC Automatic choice of standard or scientific notation.

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

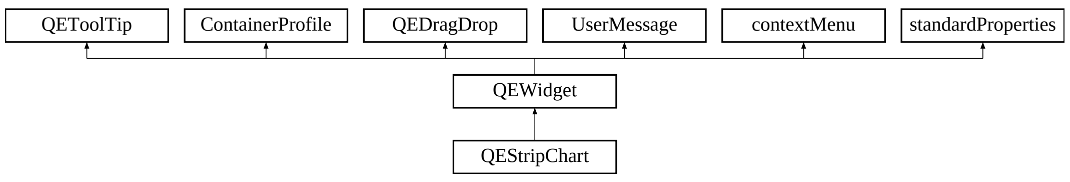
- QEStringFormatting **stringFormatting**

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)

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

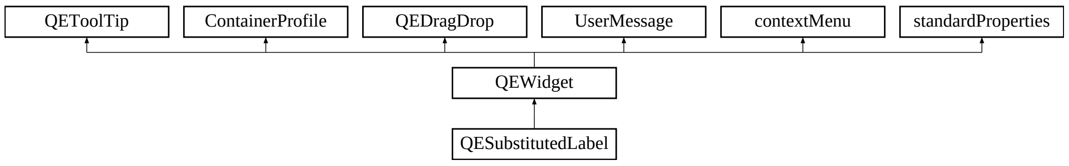
- **QEStripChartTimeDialog** (QWidget ∗parent=0)
- void **setMaximumDateTime** (QDateTime datetime)
- void **setStartDateTime** (QDateTime datetime)
- QDateTime **getStartDateTime** ()
- void **setEndDateTime** (QDateTime datetime)
- QDateTime **getEndDateTime** ()

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]`

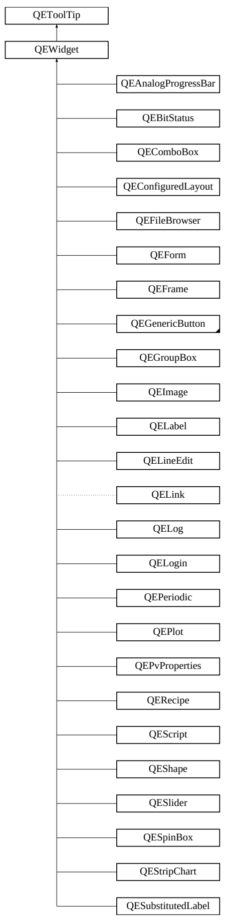
Text substitutions. These substitutions are applied to the 'labelText' property prior to copying it to the label text.

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

- **QEToolTip** (QWidget ∗ownerIn)
- void **updateToolTipVariable** (const QString &variable)
- void **updateToolTipAlarm** (const QString &alarm)
- void **updateToolTipConnection** (bool connection)

Protected Member Functions

- void **setVariableAsToolTip** (bool variableAsToolTip)
- bool **getVariableAsToolTip** ()

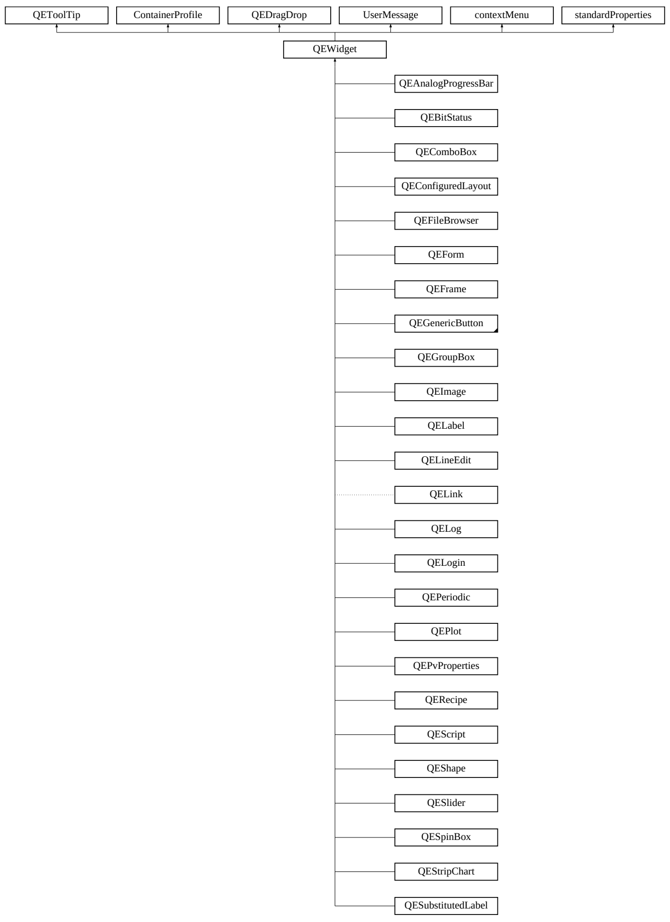
The documentation for this class was generated from the following files:

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

9.97 QEWidget Class Reference

`#include <QEWidget.h>`

Inheritance diagram for QEWidget:



Public Member Functions

- QEWidget (QWidget ∗ownerIn)

  Constructor.
- virtual ∼QEWidget ()

  Destructor.
- 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)

Protected Attributes

- bool **subscribe**

9.97.1 Detailed Description

This class is used as a base for all CA aware wigets, such as QELabel, QESpinBox, etc. It manages common issues including creating a source of CA data updates, handling error, warning and status messages, and setting tool tips based on variable names.

Note, there is tight integration between the CA aware widget classes, this class, and its base classes, especially VariableNameManager and QEToolTip.

In particular, this class manages QCaObject classes that stream updates to the CA aware widget class. But this class, however, doesn't know how to format the data, or how the updates will be used. To resolve this, this class asks its parent class (such as QELabel) to create the QCaObject class in what ever flavour it wants, by calling the virtual function createQcaItem. A QELabel, for example, wants string updates so it creates a QEString which is based on a QCaObject class and formats all updates as strings.

The CA aware parent class (such as QELabel) defines a variable by calling Variable-NameManager::setVariableName(). The VariableNamePropertyManager class calls the establishConnection function of the CA aware parent class, such as QELabel when it has a new variable name.

This class uses its base QEToolTip class to format tool tips. that class in turn calls the CA aware parent class (such as QELabel) directly to make use of a new tool tip.

After construction, a CA aware widget is activated (starts updating) by calling it's establishConnection() function in one of two ways:

1) The variable name or variable name substitutions is changed by calling setVariable-Name or setVariableNameSubstitutions respectively. These functions are in the -VariableNameManager class. The VariableNamePropertyManager calls a virtual function establishConnection() which is implemented by the CA aware widget. This is how a CA aware widget is activated in 'designer'. It occurs when 'designer' updates the variable name property or variable name substitution property.

2) When an QEForm widget is created, resulting in a set of CA aware widgets being created by loading a UI file contining plugin definitions. After loading the plugin widgets, code in the QEForm class calls the activate() function in this class (QEWiget). the activate() function calls establishConnection() in the CA aware widget for each variable. This simulates what the VariableNamePropertyManager does as each variable name is entered (see 1, above, for details)

No matter which way a CA aware widget is activated, the establishConnection() function in the CA aware widget is called for each variable. The establishConnection() function asks this QEWidget base class, by calling the createConnection() function, to perform the tasks common to all CA aware widgets for establishing a stream of CA data.

The createConnection() function sets up the widget 'tool tip', then immedietly calls the CA aware widget back asking it to create an object based on QCaObject. This object will supply a stream of CA update signals to the CA aware object in a form that it needs. For example a QELabel creates a QEString object. The QEString class is based on the QCaObject class and converts all update data to a strings which is required for updating a Qt label widget. This class stores the QCaObject based class.

After the establishConnection() function in the CA aware widget has called create-Connection(), the remaining task of the establishConnection() function is to connect the signals of the newly created QCaObject based classes to its own slots so that data updates can be used. For example, a QELabel connects the 'stringChanged' signal fromthe QEString object to its setLabelText slot.

9.97.2 Member Function Documentation

9.97.2.1 void QEWidget::activate ( )

Initiate updates. Called after all configuration is complete.

9.97.2.2 QString QEWidget::defaultFileLocation ( )

Returns the default location to create files. Use this to create files in a consistant location

9.97.2.3 QColor QEWidget::getColor (QCaAlarmInfo & *alarmInfo*, const int *saturation*)

Return a colour to update the widget's look to reflect the current alarm state Note, the color is determined by the alarmInfo class, but since that class is used in non gui applications, it can't return a QColor

9.97.2.4 unsigned int QEWidget::getMessageSourceId ( ) [inline]

Get the message source ID. The message source ID is used as part of the system where QE widgets can emit a message and have the right QE widget in the right form catch the message. Refer to the UserMessage class for further details.

9.97.2.5 qcaobject::QCaObject ∗ QEWidget::getQcaItem (unsigned int *variableIndex*)

Return a reference to one of the qCaObjects used to stream CA updates

9.97.2.6 QFile ∗ QEWidget::openQEFile (QString *name,* QFile::OpenModeFlag *mode*)

Looks for a file in a standard set of locations (and opens the file)

9.97.2.7 void QEWidget::readNow ( )

Perform a single shot read on all variables (Usefull when not subscribing by default)

9.97.2.8 void QEWidget::setMessageSourceId (unsigned int *messageSourceId*) [inline]

Set the message source ID. The message source ID is used as part of the system where QE widgets can emit a message and have the right QE widget in the right form catch the message. Refer to the UserMessage class for further details.

9.97.2.9 void QEWidget::setupContextMenu (QWidget ∗ *w*)

Take a menu widgt and add it as the context menu for this widget

9.97.2.10 void QEWidget::setVariableNameAndSubstitutions (QString *variableNameIn,* QString *variableNameSubstitutionsIn,* unsigned int *variableIndex*) [virtual]

Virtual function that may be implimented by users of QEWidget to update variable names and macro substitutions. A default is provided that is suitible in most cases.

Reimplemented in QEAnalogProgressBar, QEBitStatus, and QEForm.

9.97.2.11 virtual void QEWidget::writeNow ( ) [inline, virtual]

(Control widgets only - such as QELineEdit) Write the value now. Used when writeOnChange, writeOnEnter, etc are all false

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

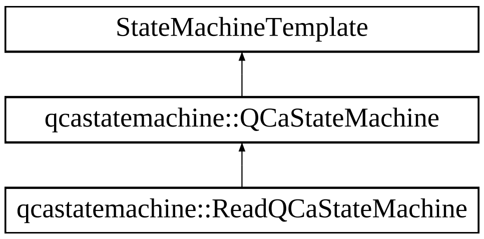
- **QEWidgets** (QObject ∗parent=0)
- virtual QList < QDesignerCustomWidgetInterface ∗ > **customWidgets** () const

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:


StateMachineTemplate
qcastatemachine::QCaStateMachine
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)
- void **appendOrReplace** (RecordSpec ∗pRecordSpec)

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

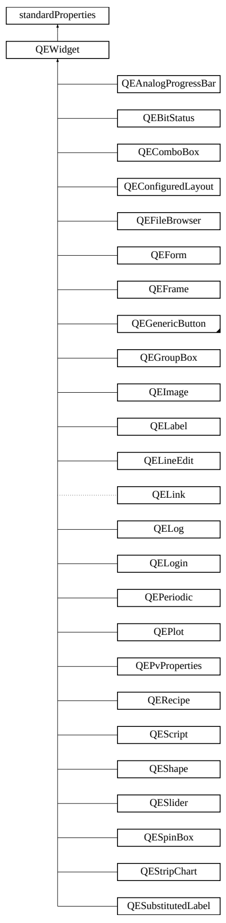
- **selectMenu** (QWidget ∗parent=0)
- imageContextMenu::imageContextMenuOptions **getSelectOption** (const QPoint &pos)
- void **setChecked** (const int mode)
- void **setPanEnabled** (bool enablePan)
- void **setVSliceEnabled** (bool enableVSliceSelection)
- void **setHSlicetEnabled** (bool enableHSliceSelection)
- 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

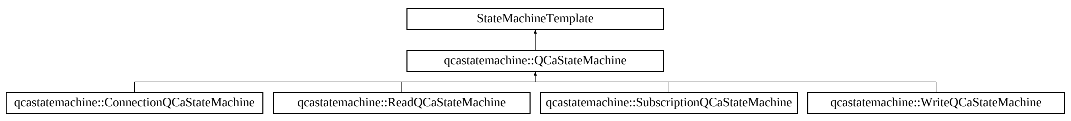
- userLevels **getUserLevelVisibility** ()
- void **setUserLevelVisibility** (userLevels level)
- userLevels **getUserLevelEnabled** ()
- void **setUserLevelEnabled** (userLevels level)
- bool **getApplicationEnabled** () const
- void **setApplicationEnabled** (bool state)
- void **setDataDisabled** (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:


StateMachineTemplate
qcastatemachine::QCaStateMachine
qcastatemachine::ConnectionQCaStateMachine
qcastatemachine::ReadQCaStateMachine
qcastatemachine::SubscriptionQCaStateMachine
qcastatemachine::WriteQCaStateMachine

Public Member Functions

- virtual bool **process** (int requestedState)=0

Public Attributes

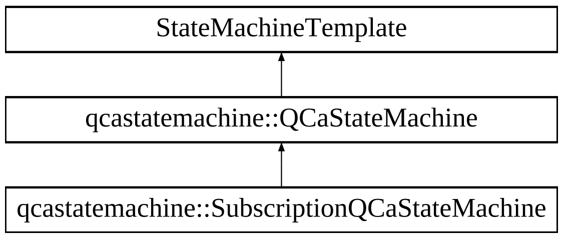
- int **currentState**
- int **requestState**

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:


StateMachineTemplate
qcastatemachine::QCaStateMachine
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)

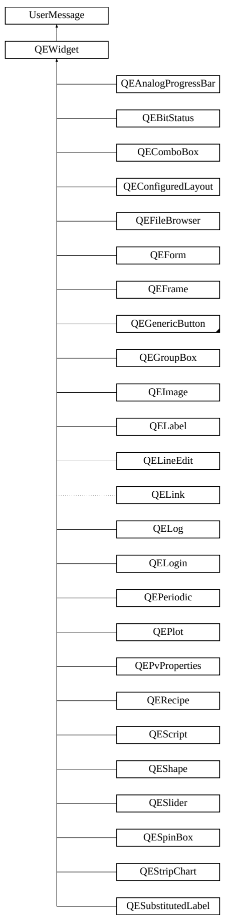
The documentation for this class was generated from the following files:

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

9.112 UserMessage Class Reference

```
#include <UserMessage.h>
```

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)

  Set the source ID (the ID set up by the GUI designer, usually matched to the source ID of logging widgets)
- void setFormId (unsigned int formId)

  Set the form ID (the the same ID for all sibling widgets within an QEForm widget)
- void setFormFilter (message_filter_options formFilterIn)

  Set the message filtering applied to the form ID.
- void setSourceFilter (message_filter_options sourceFilterIn)

  Set the message filtering applied to the source ID.

- unsigned int getSourceId ()

  Get the source ID (the ID set up by the GUI designer, usually matched to the source ID of logging widgets.
- unsigned int getFormId ()

  Get the form ID (the the same ID for all sibling widgets within an QEForm widget)
- message_filter_options getFormFilter ()

  Get the message filtering applied to the form ID.
- message_filter_options getSourceFilter ()

  Get the message filtering applied to the source ID.
- void setChildFormId (unsigned int)

  Set the for ID of all widgets that are children of this widget.
- unsigned int getChildFormId ()

  Get the for ID of all widgets that are children of this widget.
- unsigned int getNextMessageFormId ()

  Generate a new form ID for all widgets in a new form.
- void sendMessage (QString message, message_types type=MESSAGE_TYPE-_INFO)

  Send a message to the user.
- void sendMessage (QString message, QString source, message_types type=M-ESSAGE_TYPE_INFO)

  Send a message to the user with a source reference.
- QString getMessageTypeName (message_types type)

  Convenience function to provide string names for each message type.
- virtual void newMessage (QString, message_types)

  Virtual function to pass messages to derived classes (typicaly logging widgets or application windows)

Friends

- class **UserMessageSlot**
- class **UserMessageSignal**

9.112.1 Detailed Description

A class to manage user messages.

This class passes messages between widgets and application code

This class is used as a base class.

Messages are sent by calling sendMessage() Messages are received by implementing newMessage() in the derived class.

Messages can be filtered based on a source ID or a form ID

The derived widget is free to set the source ID to any value

Derived form widgets (QEForm) get a unique form ID using getNextMessageFormId() (as well as being able to set a source ID like any other QE widget) and pass this unique form ID to all widgets within the form using the ContainerProfile class.

Messages sent by a QE widget are received by all QE widgets and can filter the messages required by form ID and source ID. The form ID is under the management of the QEForm widget, the source ID is under the control of the GUI designer.

The QEForm widget does not display messages, but re-send them using its own form ID. Read on to see how this can be used.

Widgets that generate messages, and widgets (or application code) that use messages can be set up as follows:

- Application wide logging: An application with a single log window can can base a class on the UserMessage class and set up filtering to receive all messages. An application with log messages for seperate windows containing QEForm widgets (such as QEGui) can base each window class on the UserMessage class, then set up filtering for the appropriate form ID.

- Logging within a QEForm. A logging widget can be set to filter matching on the current form and so will pick up messages from any sibling widget. This includes messages from a sibling widget which is a nested QEForm. Whatever messages that nested form is set to receive, it will resend to its siblings. For example, if it is set to receive messages from the widgets it contains, these are resent up one level to the main form. If messages are dealt with within the nested QEForm (for example, it may have its own logging QE widget) then the nested QEForm could be set up not to filter and resend any messages.

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

```
#include <UserMessage.h>
```

Signals

- void message (QString msg, message_types type, unsigned int formId, unsigned int sourceId, UserMessage ∗originator)

  Emit a message signal. Any widget based on the UserMessage class can recieve these messages, filtered on formId and sourceId.

Public Member Functions

- void sendMessage (QString msg, message_types type, unsigned int formId, unsigned int sourceId, UserMessage ∗originator)

  Send a message to all widgets based on the UserMessage class.

9.113.1 Detailed Description

Class used to send message signals. Used only within UserMessage.cpp A single instance of this class is shared by all instances of the UserMessage class. This allows every UserMessage class instance to connect to a single source of messages

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

```
#include <UserMessage.h>
```

Public Slots

- void message (QString msg, message_types type, unsigned int formId, unsigned int sourceId, UserMessage ∗originator)

  A message has been received.

Public Member Functions

- void setOwner (UserMessage ∗ownerIn)

  Set the UserMessage class this is a part of.

9.114.1 Detailed Description

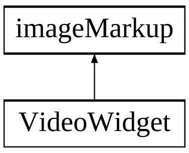
Class used to receive message signals. Used only within UserMessage.cpp An instance of this class is created by all instances of the UserMessage class. The UserMessage class uses an instance of this class to receive messages so it does not have to be based on QObject itself. This is required as derived classes generally need to be also based on another object derived from QObject (and QObject can only be the base of a single base class)

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, bool clearing, 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, bool clearing, 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

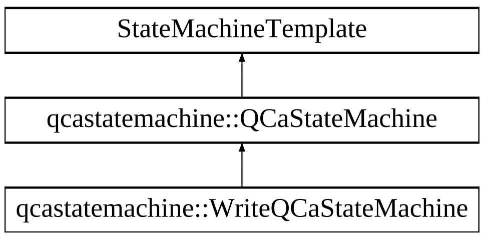
- **WidgetRef** (QEWidget ∗refIn)
- QEWidget ∗ **getRef** ()

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:


StateMachineTemplate
qcastatemachine::QCaStateMachine
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