

EPICS QT Framework

2.8.0

Generated by Doxygen 1.7.4

Tue Apr 22 2014 14:57:24

Contents

1 QE framework - EPICS aware Qt Widgets and data access classes	1
1.1 Documentation	1
1.2 License	2
1.3 Platforms	2
1.4 Screenshots	2
1.5 Downloads	2
1.6 Installation	2
1.7 Support	3
1.8 Related Projects	3
1.9 Credits:	3
2 GNU General Public License	5
3 ASgui screen shots	7
4 other applications using epicsqt widgets	13
5 Qt Designer	15
6 Qt Creator	17
7 Class Index	19
7.1 Class Hierarchy	19
8 Class Index	25
8.1 Class List	25
9 Class Documentation	29
9.1 _CopyPaste Class Reference	29

9.2	_Field Class Reference	29
9.3	_Item Class Reference	30
9.4	_QDialogItem Class Reference	31
9.5	_QPushButtonGroup Class Reference	31
9.6	_QTableWidgetFileBrowser Class Reference	31
9.7	_QTableWidgetLog Class Reference	32
9.8	_QTableWidgetScript Class Reference	32
9.9	applicationLauncher Class Reference	32
9.10	arealInfo Class Reference	33
9.11	QEAnalogIndicator::Band Struct Reference	34
9.12	QEAnalogIndicator::BandList Class Reference	34
9.13	qcastatemachine::ConnectionQCaStateMachine Class Reference	34
9.14	ContainerProfile Class Reference	35
9.15	contextMenu Class Reference	36
9.16	contextMenuObject Class Reference	38
9.17	QEPeriodic::elementInfoStruct Struct Reference	39
9.18	FFBuffer Class Reference	39
9.19	FFThread Class Reference	40
9.20	flipRotateMenu Class Reference	40
9.21	fullScreenWindow Class Reference	40
9.22	histogram Class Reference	41
9.23	histogramScroll Class Reference	41
9.24	historicImage Class Reference	41
9.25	imageContextMenu Class Reference	42
9.26	imageDisplayProperties Class Reference	43
9.27	imageInfo Class Reference	44
9.28	imageMarkup Class Reference	45
9.29	imageUpdateIndicator Class Reference	47
9.30	loginWidget Class Reference	47
9.31	managePixmaps Class Reference	48
9.32	markupBeam Class Reference	48
9.33	markupDisplayMenu Class Reference	49
9.34	markupEllipse Class Reference	49
9.35	markupHLine Class Reference	50

9.35.1 Member Function Documentation	51
9.35.1.1 drawMarkup	51
9.36 markupItem Class Reference	51
9.37 markupLine Class Reference	53
9.38 markupRegion Class Reference	54
9.39 markupTarget Class Reference	55
9.40 markupText Class Reference	55
9.41 markupVLine Class Reference	56
9.41.1 Member Function Documentation	57
9.41.1.1 drawMarkup	57
9.42 message_types Class Reference	57
9.43 mpegSource Class Reference	57
9.43.1 Member Function Documentation	58
9.43.1.1 updateImage	58
9.44 mpegSourceObject Class Reference	58
9.45 QEStripChartToolBar::OwnWidgets Class Reference	59
9.46 PeriodicDialog Class Reference	59
9.47 PeriodicElementSetupForm Class Reference	60
9.48 PeriodicSetupDialog Class Reference	60
9.49 PersistanceManager Class Reference	60
9.50 playbackTimer Class Reference	61
9.51 PMContext Class Reference	61
9.52 PMElement Class Reference	62
9.53 PMElementList Class Reference	62
9.53.1 Member Function Documentation	63
9.53.1.1 getElement	63
9.54 pointInfo Class Reference	63
9.55 processManager Class Reference	63
9.56 profilePlot Class Reference	64
9.57 PublishedProfile Class Reference	64
9.58 QBitStatus Class Reference	65
9.59 QCaAlarmInfo Class Reference	67
9.60 QCaConnectionInfo Class Reference	67
9.61 QCaDataPoint Class Reference	68

9.62 QCaDataPointList Class Reference	68
9.63 QCaDateTime Class Reference	69
9.63.1 Member Function Documentation	69
9.63.1.1 floating	69
9.64 QCaEventFilter Class Reference	69
9.65 QCaEventItem Class Reference	70
9.66 QCaEventUpdate Class Reference	70
9.67 QCaInstalledFiltersListItem Class Reference	70
9.68 qcaobject::QCaObject Class Reference	71
9.69 qcastatemachine::QCaStateMachine Class Reference	73
9.70 QCaVariableNamePropertyManager Class Reference	73
9.71 QEAnalogIndicator Class Reference	74
9.71.1 Detailed Description	77
9.71.2 Member Enumeration Documentation	77
9.71.2.1 Modes	77
9.71.2.2 Orientations	77
9.71.3 Property Documentation	77
9.71.3.1 backgroundColour	77
9.71.3.2 borderColour	77
9.71.3.3 centreAngle	78
9.71.3.4 fontColour	78
9.71.3.5 foregroundColour	78
9.71.3.6 logScale	78
9.71.3.7 logScaleInterval	78
9.71.3.8 majorInterval	78
9.71.3.9 maximum	78
9.71.3.10 minimum	78
9.71.3.11 minorInterval	78
9.71.3.12 mode	78
9.71.3.13 orientation	79
9.71.3.14 showScale	79
9.71.3.15 showText	79
9.71.3.16 spanAngle	79
9.71.3.17 value	79

9.72 QEAnalogProgressBar Class Reference	79
9.72.1 Member Enumeration Documentation	82
9.72.1.1 ArrayActions	82
9.72.1.2 Formats	82
9.72.1.3 Notations	83
9.72.1.4 UserLevels	83
9.72.2 Constructor & Destructor Documentation	83
9.72.2.1 QEAnalogProgressBar	83
9.72.2.2 QEAnalogProgressBar	83
9.72.3 Member Function Documentation	83
9.72.3.1 dbValueChanged	83
9.72.4 Property Documentation	84
9.72.4.1 addUnits	84
9.72.4.2 alarmSeverityDisplayStyle	84
9.72.4.3 allowDrop	84
9.72.4.4 arrayAction	84
9.72.4.5 displayAlarmState	84
9.72.4.6 format	84
9.72.4.7 int	85
9.72.4.8 leadingZero	85
9.72.4.9 localEnumeration	85
9.72.4.10 notation	86
9.72.4.11 precision	86
9.72.4.12 trailingZeros	86
9.72.4.13 useDbDisplayLimits	86
9.72.4.14 useDbPrecision	86
9.72.4.15 userLevelEnabled	86
9.72.4.16 userLevelEngineerStyle	86
9.72.4.17 userLevelScientistStyle	87
9.72.4.18 userLevelUserStyle	87
9.72.4.19 userLevelVisibility	87
9.72.4.20 variable	87
9.72.4.21 variableAsToolTip	87
9.72.4.22 variableSubstitutions	87

9.72.4.23	visible	88
9.73	QEBitStatus Class Reference	88
9.73.1	Member Enumeration Documentation	89
9.73.1.1	UserLevels	89
9.73.2	Member Function Documentation	90
9.73.2.1	dbValueChanged	90
9.73.2.2	setVariableNameAndSubstitutions	90
9.73.3	Property Documentation	90
9.73.3.1	allowDrop	90
9.73.3.2	displayAlarmState	90
9.73.3.3	int	90
9.73.3.4	userLevelEnabled	91
9.73.3.5	userLevelEngineerStyle	91
9.73.3.6	userLevelScientistStyle	91
9.73.3.7	userLevelUserStyle	91
9.73.3.8	userLevelVisibility	91
9.73.3.9	variable	92
9.73.3.10	variableAsToolTip	92
9.73.3.11	variableSubstitutions	92
9.73.3.12	visible	92
9.74	QEByteArray Class Reference	92
9.75	QECheckBox Class Reference	93
9.75.1	Member Enumeration Documentation	96
9.75.1.1	ArrayActions	96
9.75.1.2	CreationOptionNames	97
9.75.1.3	Formats	97
9.75.1.4	Notations	97
9.75.1.5	ProgramStartupOptionNames	98
9.75.1.6	UpdateOptions	98
9.75.1.7	UserLevels	98
9.75.2	Constructor & Destructor Documentation	99
9.75.2.1	QECheckBox	99
9.75.2.2	QECheckBox	99
9.75.3	Member Function Documentation	99

9.75.3.1	clicked	99
9.75.3.2	dbValueChanged	99
9.75.3.3	pressed	99
9.75.3.4	released	99
9.75.3.5	requestAction	99
9.75.4	Property Documentation	100
9.75.4.1	addUnits	100
9.75.4.2	alignment	100
9.75.4.3	allowDrop	100
9.75.4.4	arguments	100
9.75.4.5	arrayAction	100
9.75.4.6	clickCheckedText	101
9.75.4.7	clickText	101
9.75.4.8	confirmAction	101
9.75.4.9	confirmText	101
9.75.4.10	creationOption	101
9.75.4.11	customisationName	101
9.75.4.12	displayAlarmState	102
9.75.4.13	format	102
9.75.4.14	guiFile	102
9.75.4.15	int	102
9.75.4.16	labelText	102
9.75.4.17	leadingZero	103
9.75.4.18	localEnumeration	103
9.75.4.19	notation	103
9.75.4.20	password	103
9.75.4.21	pixmap0	104
9.75.4.22	pixmap1	104
9.75.4.23	pixmap2	104
9.75.4.24	pixmap3	104
9.75.4.25	pixmap4	104
9.75.4.26	pixmap5	104
9.75.4.27	pixmap6	104
9.75.4.28	pixmap7	104

9.75.4.29 precision	105
9.75.4.30 pressText	105
9.75.4.31 prioritySubstitutions	105
9.75.4.32 program	105
9.75.4.33 programStartupOption	105
9.75.4.34 releaseText	105
9.75.4.35 subscribe	105
9.75.4.36 trailingZeros	106
9.75.4.37 updateOption	106
9.75.4.38 useDbPrecision	106
9.75.4.39 userLevelEnabled	106
9.75.4.40 userLevelEngineerStyle	106
9.75.4.41 userLevelScientistStyle	106
9.75.4.42 userLevelUserStyle	107
9.75.4.43 userLevelVisibility	107
9.75.4.44 variable	107
9.75.4.45 variableAsToolTip	107
9.75.4.46 variableSubstitutions	107
9.75.4.47 visible	107
9.75.4.48 writeOnClick	107
9.75.4.49 writeOnPress	108
9.75.4.50 writeOnRelease	108
9.76 QECheckBoxManager Class Reference	108
9.77 QEComboBox Class Reference	108
9.77.1 Member Enumeration Documentation	110
9.77.1.1 UserLevels	110
9.77.2 Member Function Documentation	111
9.77.2.1 dbValueChanged	111
9.77.3 Member Data Documentation	111
9.77.3.1 useDbEnumerations	111
9.77.3.2 writeOnChange	111
9.77.4 Property Documentation	111
9.77.4.1 allowDrop	111
9.77.4.2 displayAlarmState	111

9.77.4.3 int	111
9.77.4.4 localEnumeration	112
9.77.4.5 subscribe	112
9.77.4.6 userLevelEnabled	112
9.77.4.7 userLevelEngineerStyle	112
9.77.4.8 userLevelScientistStyle	112
9.77.4.9 userLevelUserStyle	112
9.77.4.10 userLevelVisibility	113
9.77.4.11 variable	113
9.77.4.12 variableAsToolTip	113
9.77.4.13 variableSubstitutions	113
9.77.4.14 visible	113
9.78 QEConfiguredLayout Class Reference	113
9.79 QEConfiguredLayoutManager Class Reference	115
9.80 QEDragDrop Class Reference	116
9.81 QEFileBrowser Class Reference	117
9.81.1 Detailed Description	119
9.81.2 Member Enumeration Documentation	120
9.81.2.1 UserLevels	120
9.81.3 Member Function Documentation	120
9.81.3.1 selected	120
9.81.4 Property Documentation	120
9.81.4.1 allowDrop	120
9.81.4.2 displayAlarmState	120
9.81.4.3 int	120
9.81.4.4 userLevelEnabled	121
9.81.4.5 userLevelEngineerStyle	121
9.81.4.6 userLevelScientistStyle	121
9.81.4.7 userLevelUserStyle	121
9.81.4.8 userLevelVisibility	121
9.81.4.9 variable	122
9.81.4.10 variableAsToolTip	122
9.81.4.11 variableSubstitutions	122
9.81.4.12 visible	122

9.82 QEFloating Class Reference	122
9.83 QEFloatingArray Class Reference	123
9.83.1 Detailed Description	123
9.84 QEFloatingFormatting Class Reference	124
9.85 QEForm Class Reference	124
9.86 QEFrame Class Reference	126
9.86.1 Member Enumeration Documentation	127
9.86.1.1 UserLevels	127
9.86.2 Property Documentation	127
9.86.2.1 allowDrop	127
9.86.2.2 displayAlarmState	127
9.86.2.3 int	127
9.86.2.4 userLevelEnabled	128
9.86.2.5 userLevelEngineerStyle	128
9.86.2.6 userLevelScientistStyle	128
9.86.2.7 userLevelUserStyle	128
9.86.2.8 userLevelVisibility	128
9.86.2.9 variableAsToolTip	129
9.86.2.10 visible	129
9.87 QEGenericButton Class Reference	129
9.88 QEGenericEdit Class Reference	131
9.88.1 Member Enumeration Documentation	133
9.88.1.1 UserLevels	133
9.88.2 Constructor & Destructor Documentation	133
9.88.2.1 QEGenericEdit	133
9.88.2.2 QEGenericEdit	134
9.88.3 Member Function Documentation	134
9.88.3.1 getConfirmWrite	134
9.88.3.2 getSubscribe	134
9.88.3.3 getWriteOnEnter	134
9.88.3.4 getWriteOnFinish	134
9.88.3.5 getWriteOnLoseFocus	134
9.88.3.6 setConfirmWrite	134
9.88.3.7 setSubscribe	134

9.88.3.8	setWriteOnEnter	135
9.88.3.9	setWriteOnFinish	135
9.88.3.10	setWriteOnLoseFocus	135
9.88.3.11	writeNow	135
9.88.4	Property Documentation	135
9.88.4.1	allowDrop	135
9.88.4.2	confirmWrite	135
9.88.4.3	displayAlarmState	135
9.88.4.4	int	136
9.88.4.5	subscribe	136
9.88.4.6	userLevelEnabled	136
9.88.4.7	userLevelEngineerStyle	136
9.88.4.8	userLevelScientistStyle	136
9.88.4.9	userLevelUserStyle	137
9.88.4.10	userLevelVisibility	137
9.88.4.11	variable	137
9.88.4.12	variableAsToolTip	137
9.88.4.13	variableSubstitutions	137
9.88.4.14	visible	137
9.88.4.15	writeOnEnter	137
9.88.4.16	writeOnFinish	138
9.88.4.17	writeOnLoseFocus	138
9.89	QEGroupBox Class Reference	138
9.89.1	Member Enumeration Documentation	139
9.89.1.1	UserLevels	139
9.89.2	Property Documentation	140
9.89.2.1	allowDrop	140
9.89.2.2	displayAlarmState	140
9.89.2.3	int	140
9.89.2.4	substitutedTitle	140
9.89.2.5	textSubstitutions	140
9.89.2.6	userLevelEnabled	140
9.89.2.7	userLevelEngineerStyle	141
9.89.2.8	userLevelScientistStyle	141

9.89.2.9	userLevelUserStyle	141
9.89.2.10	userLevelVisibility	141
9.89.2.11	variableAsToolTip	141
9.89.2.12	visible	142
9.90	QEImage Class Reference	142
9.90.1	Member Enumeration Documentation	157
9.90.1.1	FormatOptions	157
9.90.1.2	ProgramStartupOptionNames	158
9.90.1.3	resizeOptions	158
9.90.1.4	ResizeOptions	158
9.90.1.5	RotationOptions	158
9.90.1.6	rotationOptions	159
9.90.1.7	selectOptions	159
9.90.1.8	UserLevels	159
9.90.2	Constructor & Destructor Documentation	159
9.90.2.1	QEImage	159
9.90.2.2	QEImage	160
9.90.3	Member Function Documentation	160
9.90.3.1	dbValueChanged	160
9.90.4	Member Data Documentation	160
9.90.4.1	displayButtonBar	160
9.90.4.2	initialVertScrollPos	160
9.90.5	Property Documentation	160
9.90.5.1	allowDrop	160
9.90.5.2	areaColor	160
9.90.5.3	arguments1	161
9.90.5.4	arguments2	161
9.90.5.5	autoBrightnessContrast	161
9.90.5.6	beamColor	161
9.90.5.7	beamXVariable	161
9.90.5.8	beamYVariable	161
9.90.5.9	bitDepthVariable	161
9.90.5.10	briefInfoArea	161
9.90.5.11	clippingHighVariable	161

9.90.5.12 clippingLowVariable	162
9.90.5.13 clippingOnOffVariable	162
9.90.5.14 contrastReversal	162
9.90.5.15 dimension1Variable	162
9.90.5.16 dimension2Variable	162
9.90.5.17 dimension3Variable	162
9.90.5.18 dimensionsVariable	162
9.90.5.19 displayAlarmState	162
9.90.5.20 displayArea1Selection	163
9.90.5.21 displayArea2Selection	163
9.90.5.22 displayArea3Selection	163
9.90.5.23 displayArea4Selection	163
9.90.5.24 displayBeamSelection	163
9.90.5.25 displayCursorPixelInfo	163
9.90.5.26 displayEllipse	163
9.90.5.27 displayHozSliceSelection	163
9.90.5.28 displayProfileSelection	163
9.90.5.29 displayTargetSelection	164
9.90.5.30 displayVertSliceSelection	164
9.90.5.31 ellipseColor	164
9.90.5.32 ellipseX1Variable	164
9.90.5.33 ellipseX2Variable	164
9.90.5.34 ellipseY1Variable	164
9.90.5.35 ellipseY2Variable	164
9.90.5.36 enableArea1Selection	164
9.90.5.37 enableArea2Selection	164
9.90.5.38 enableArea3Selection	165
9.90.5.39 enableArea4Selection	165
9.90.5.40 enableBeamSelection	165
9.90.5.41 enableHozSliceSelection	165
9.90.5.42 enableProfileSelection	165
9.90.5.43 enableTargetSelection	165
9.90.5.44 enableVertSliceSelection	165
9.90.5.45 externalControls	165

9.90.5.46 formatOption	166
9.90.5.47 formatVariable	166
9.90.5.48 heightVariable	166
9.90.5.49 horizontalFlip	166
9.90.5.50 hozSliceColor	166
9.90.5.51 imageVariable	166
9.90.5.52 initialHosScrollPos	166
9.90.5.53 int	166
9.90.5.54 lineProfileArrayVariable	167
9.90.5.55 lineProfileThicknessVariable	167
9.90.5.56 lineProfileX1Variable	167
9.90.5.57 lineProfileX2Variable	167
9.90.5.58 lineProfileY1Variable	167
9.90.5.59 lineProfileY2Variable	167
9.90.5.60 logBrightness	167
9.90.5.61 profileColor	167
9.90.5.62 profileHozArrayVariable	168
9.90.5.63 profileHozThicknessVariable	168
9.90.5.64 profileHozVariable	168
9.90.5.65 profileVertArrayVariable	168
9.90.5.66 profileVertThicknessVariable	168
9.90.5.67 profileVertVariable	168
9.90.5.68 program1	168
9.90.5.69 program2	168
9.90.5.70 programStartupOption1	169
9.90.5.71 programStartupOption2	169
9.90.5.72 regionOfInterest1HVariable	169
9.90.5.73 regionOfInterest1WVariable	169
9.90.5.74 regionOfInterest1XVariable	169
9.90.5.75 regionOfInterest1YVariable	169
9.90.5.76 regionOfInterest2HVariable	169
9.90.5.77 regionOfInterest2WVariable	169
9.90.5.78 regionOfInterest2XVariable	170
9.90.5.79 regionOfInterest2YVariable	170

9.90.5.80 regionOfInterest3HVariable	170
9.90.5.81 regionOfInterest3WVariable	170
9.90.5.82 regionOfInterest3XVariable	170
9.90.5.83 regionOfInterest3YVariable	170
9.90.5.84 regionOfInterest4HVariable	170
9.90.5.85 regionOfInterest4WVariable	170
9.90.5.86 regionOfInterest4XVariable	170
9.90.5.87 regionOfInterest4YVariable	171
9.90.5.88 resizeOption	171
9.90.5.89 rotation	171
9.90.5.90 showTime	171
9.90.5.91 targetColor	171
9.90.5.92 targetTriggerVariable	171
9.90.5.93 targetXVariable	171
9.90.5.94 targetYVariable	171
9.90.5.95 timeColor	171
9.90.5.96 URL	172
9.90.5.97 useFalseColour	172
9.90.5.98 userLevelEnabled	172
9.90.5.99 userLevelEngineerStyle	172
9.90.5.100userLevelScientistStyle	172
9.90.5.101userLevelUserStyle	172
9.90.5.102userLevelVisibility	173
9.90.5.103variableAsToolTip	173
9.90.5.104variableSubstitutions	173
9.90.5.105verticalFlip	173
9.90.5.106vertSliceColor	173
9.90.5.107visible	173
9.90.5.108widthVariable	173
9.91 QEImageMarkupThickness Class Reference	174
9.92 QEImageOptionsDialog Class Reference	174
9.93 QEInteger Class Reference	174
9.94 QEIntegerArray Class Reference	175
9.94.1 Detailed Description	175

9.95 QEIntegerFormatting Class Reference	176
9.95.1 Detailed Description	176
9.95.2 Member Function Documentation	176
9.95.2.1 formatInteger	176
9.95.2.2 formatIntegerArray	177
9.95.2.3 formatValue	177
9.96 QELabel Class Reference	177
9.96.1 Detailed Description	180
9.96.2 Member Enumeration Documentation	180
9.96.2.1 ArrayActions	180
9.96.2.2 Formats	181
9.96.2.3 Notations	181
9.96.2.4 UpdateOptions	181
9.96.2.5 updateOptions	181
9.96.2.6 UserLevels	182
9.96.3 Constructor & Destructor Documentation	182
9.96.3.1 QELabel	182
9.96.3.2 QELabel	182
9.96.4 Member Function Documentation	182
9.96.4.1 dbValueChanged	182
9.96.5 Property Documentation	182
9.96.5.1 addUnits	182
9.96.5.2 allowDrop	182
9.96.5.3 arrayAction	183
9.96.5.4 displayAlarmState	183
9.96.5.5 format	183
9.96.5.6 int	183
9.96.5.7 leadingZero	183
9.96.5.8 localEnumeration	184
9.96.5.9 notation	184
9.96.5.10 pixmap0	184
9.96.5.11 pixmap1	184
9.96.5.12 pixmap2	185
9.96.5.13 pixmap3	185

9.96.5.14 pixmap4	185
9.96.5.15 pixmap5	185
9.96.5.16 pixmap6	185
9.96.5.17 pixmap7	185
9.96.5.18 precision	185
9.96.5.19 trailingZeros	185
9.96.5.20 updateOption	185
9.96.5.21 useDbPrecision	186
9.96.5.22 userLevelEnabled	186
9.96.5.23 userLevelEngineerStyle	186
9.96.5.24 userLevelScientistStyle	186
9.96.5.25 userLevelUserStyle	186
9.96.5.26 userLevelVisibility	187
9.96.5.27 variable	187
9.96.5.28 variableAsToolTip	187
9.96.5.29 variableSubstitutions	187
9.96.5.30 visible	187
9.97 QLELineEdit Class Reference	187
9.97.1 Member Enumeration Documentation	189
9.97.1.1 ArrayActions	189
9.97.1.2 Formats	189
9.97.1.3 Notations	190
9.97.2 Constructor & Destructor Documentation	190
9.97.2.1 QLELineEdit	190
9.97.2.2 QLELineEdit	190
9.97.3 Member Function Documentation	190
9.97.3.1 dbValueChanged	190
9.97.4 Property Documentation	190
9.97.4.1 addUnits	190
9.97.4.2 arrayAction	190
9.97.4.3 format	191
9.97.4.4 int	191
9.97.4.5 leadingZero	191
9.97.4.6 localEnumeration	191

9.97.4.7 notation	192
9.97.4.8 precision	192
9.97.4.9 trailingZeros	192
9.97.4.10 useDbPrecision	192
9.98 QELineEditManager Class Reference	192
9.99 QELink Class Reference	193
9.100QELocalEnumeration Class Reference	195
9.100.1 Detailed Description	195
9.100.2 Constructor & Destructor Documentation	195
9.100.2.1 QELocalEnumeration	195
9.100.2.2 QELocalEnumeration	195
9.100.3 Member Function Documentation	196
9.100.3.1 getLocalEnumeration	196
9.100.3.2 isDefined	196
9.100.3.3 setLocalEnumeration	196
9.100.3.4 text.ToDouble	196
9.100.3.5 text.ToInt	197
9.100.3.6 textToValue	197
9.100.3.7 valueToText	197
9.101QELog Class Reference	197
9.101.1 Member Enumeration Documentation	199
9.101.1.1 UserLevels	199
9.101.2 Property Documentation	200
9.101.2.1 allowDrop	200
9.101.2.2 displayAlarmState	200
9.101.2.3 int	200
9.101.2.4 userLevelEnabled	200
9.101.2.5 userLevelEngineerStyle	200
9.101.2.6 userLevelScientistStyle	201
9.101.2.7 userLevelUserStyle	201
9.101.2.8 userLevelVisibility	201
9.101.2.9 variableAsToolTip	201
9.101.2.10 visible	201
9.102QELogin Class Reference	202

9.103QELoginDialog Class Reference	202
9.104QENumericEdit Class Reference	203
9.104.1 Detailed Description	205
9.104.2 Constructor & Destructor Documentation	205
9.104.2.1 QENumericEdit	205
9.104.2.2 QENumericEdit	205
9.104.3 Member Function Documentation	205
9.104.3.1 dbValueChanged	205
9.104.4 Property Documentation	205
9.104.4.1 addUnits	205
9.104.4.2 autoScale	205
9.104.4.3 leadingZeros	205
9.104.4.4 maximum	206
9.104.4.5 minimum	206
9.104.4.6 precision	206
9.105QENumericEditManager Class Reference	206
9.106QEPeriodic Class Reference	207
9.106.1 Member Enumeration Documentation	210
9.106.1.1 UserLevels	210
9.106.2 Member Function Documentation	210
9.106.2.1 dbElementChanged	210
9.106.2.2 dbValueChanged	210
9.106.3 Member Data Documentation	210
9.106.3.1 allowDrop	210
9.106.4 Property Documentation	210
9.106.4.1 displayAlarmState	210
9.106.4.2 int	211
9.106.4.3 readbackLabelVariable1	211
9.106.4.4 readbackLabelVariable2	211
9.106.4.5 subscribe	211
9.106.4.6 userLevelEnabled	211
9.106.4.7 userLevelEngineerStyle	211
9.106.4.8 userLevelScientistStyle	212
9.106.4.9 userLevelUserStyle	212

9.106.4.10userLevelVisibility	212
9.106.4.11variableAsToolTip	212
9.106.4.12variableSubstitutions	212
9.106.4.13visible	212
9.106.4.14writeButtonVariable1	213
9.106.4.15writeButtonVariable2	213
9.107QEPeriodicComponentData Class Reference	213
9.108QEPeriodicTaskMenu Class Reference	213
9.109QEPeriodicTaskMenuFactory Class Reference	214
9.110QEpicsPV Class Reference	214
9.111QEPlot Class Reference	215
9.111.1 Member Enumeration Documentation	218
9.111.1.1 UserLevels	218
9.111.2 Member Function Documentation	219
9.111.2.1 dbValueChanged	219
9.111.2.2 dbValueChanged	219
9.111.3 Member Data Documentation	219
9.111.3.1 allowDrop	219
9.111.4 Property Documentation	219
9.111.4.1 displayAlarmState	219
9.111.4.2 int	219
9.111.4.3 userLevelEnabled	219
9.111.4.4 userLevelEngineerStyle	220
9.111.4.5 userLevelScientistStyle	220
9.111.4.6 userLevelUserStyle	220
9.111.4.7 userLevelVisibility	220
9.111.4.8 variable1	220
9.111.4.9 variable2	221
9.111.4.10variable3	221
9.111.4.11variable4	221
9.111.4.12variableAsToolTip	221
9.111.4.13variableSubstitutions	221
9.111.4.14visible	221
9.112QEPushButton Class Reference	221

9.112.1 Member Enumeration Documentation	225
9.112.1.1 ArrayActions	225
9.112.1.2 CreationOptionNames	225
9.112.1.3 Formats	226
9.112.1.4 Notations	226
9.112.1.5 ProgramStartupOptionNames	226
9.112.1.6 UpdateOptions	226
9.112.1.7 UserLevels	227
9.112.2 Constructor & Destructor Documentation	227
9.112.2.1 QEPushButton	227
9.112.2.2 QEPushButton	227
9.112.3 Member Function Documentation	227
9.112.3.1 clicked	227
9.112.3.2 dbValueChanged	227
9.112.3.3 pressed	228
9.112.3.4 released	228
9.112.3.5 requestAction	228
9.112.4 Property Documentation	228
9.112.4.1 addUnits	228
9.112.4.2 alignment	228
9.112.4.3 allowDrop	228
9.112.4.4 altReadbackVariable	228
9.112.4.5 arguments	229
9.112.4.6 arrayAction	229
9.112.4.7 clickCheckedText	229
9.112.4.8 clickText	229
9.112.4.9 confirmAction	229
9.112.4.10confirmText	230
9.112.4.11creationOption	230
9.112.4.12customisationName	230
9.112.4.13displayAlarmState	230
9.112.4.14format	230
9.112.4.15guiFile	230
9.112.4.16nt	231

9.112.4.17labelText	231
9.112.4.18leadingZero	231
9.112.4.19localEnumeration	231
9.112.4.20notation	232
9.112.4.21password	232
9.112.4.22 pixmap0	232
9.112.4.23 pixmap1	232
9.112.4.24 pixmap2	232
9.112.4.25 pixmap3	232
9.112.4.26 pixmap4	233
9.112.4.27 pixmap5	233
9.112.4.28 pixmap6	233
9.112.4.29 pixmap7	233
9.112.4.30precision	233
9.112.4.31pressText	233
9.112.4.32prioritySubstitutions	233
9.112.4.33program	234
9.112.4.34programStartupOption	234
9.112.4.35releaseText	234
9.112.4.36subscribe	234
9.112.4.37trailingZeros	234
9.112.4.38updateOption	234
9.112.4.39useDbPrecision	234
9.112.4.40userLevelEnabled	234
9.112.4.41userLevelEngineerStyle	235
9.112.4.42userLevelScientistStyle	235
9.112.4.43userLevelUserStyle	235
9.112.4.44userLevelVisibility	235
9.112.4.45variable	235
9.112.4.46variableAsToolTip	236
9.112.4.47variableSubstitutions	236
9.112.4.48visible	236
9.112.4.49writeOnClick	236
9.112.4.50writeOnPress	236

9.112.4.5 <code>writeOnRelease</code>	236
9.113 <code>QEPVNameLists</code> Class Reference	236
9.114 <code>QEPvProperties</code> Class Reference	237
9.114.1 Member Function Documentation	238
9.114.1.1 <code>restoreConfiguration</code>	238
9.114.1.2 <code>saveConfiguration</code>	238
9.114.2 Property Documentation	238
9.114.2.1 <code>variable</code>	238
9.114.2.2 <code>variableSubstitutions</code>	238
9.115 <code>QEPvPropertiesManager</code> Class Reference	239
9.116 <code>QERadioButton</code> Class Reference	239
9.116.1 Member Enumeration Documentation	242
9.116.1.1 <code>ArrayActions</code>	242
9.116.1.2 <code>CreationOptionNames</code>	243
9.116.1.3 <code>Formats</code>	243
9.116.1.4 <code>Notations</code>	243
9.116.1.5 <code>ProgramStartupOptionNames</code>	244
9.116.1.6 <code>UpdateOptions</code>	244
9.116.1.7 <code>UserLevels</code>	244
9.116.2 Constructor & Destructor Documentation	245
9.116.2.1 <code>QERadioButton</code>	245
9.116.2.2 <code>QERadioButton</code>	245
9.116.3 Member Function Documentation	245
9.116.3.1 <code>clicked</code>	245
9.116.3.2 <code>dbValueChanged</code>	245
9.116.3.3 <code>pressed</code>	245
9.116.3.4 <code>released</code>	245
9.116.3.5 <code>requestAction</code>	245
9.116.4 Property Documentation	246
9.116.4.1 <code>addUnits</code>	246
9.116.4.2 <code>alignment</code>	246
9.116.4.3 <code>allowDrop</code>	246
9.116.4.4 <code>arguments</code>	246
9.116.4.5 <code>arrayAction</code>	246

9.116.4.6 clickCheckedText	247
9.116.4.7 clickText	247
9.116.4.8 confirmAction	247
9.116.4.9 confirmText	247
9.116.4.10creationOption	247
9.116.4.11customisationName	247
9.116.4.12displayAlarmState	248
9.116.4.13format	248
9.116.4.14guiFile	248
9.116.4.15nt	248
9.116.4.16labelText	248
9.116.4.17leadingZero	249
9.116.4.18localEnumeration	249
9.116.4.19notation	249
9.116.4.20password	249
9.116.4.21 pixmap0	250
9.116.4.22 pixmap1	250
9.116.4.23 pixmap2	250
9.116.4.24 pixmap3	250
9.116.4.25 pixmap4	250
9.116.4.26 pixmap5	250
9.116.4.27 pixmap6	250
9.116.4.28 pixmap7	250
9.116.4.29 precision	251
9.116.4.30 pressText	251
9.116.4.31 prioritySubstitutions	251
9.116.4.32 program	251
9.116.4.33 programStartupOption	251
9.116.4.34 releaseText	251
9.116.4.35 subscribe	251
9.116.4.36 trailingZeros	252
9.116.4.37 updateOption	252
9.116.4.38 useDbPrecision	252
9.116.4.39 userLevelEnabled	252

9.116.4.40 userLevelEngineerStyle	252
9.116.4.41 userLevelScientistStyle	252
9.116.4.42 userLevelUserStyle	253
9.116.4.43 userLevelVisibility	253
9.116.4.44 variable	253
9.116.4.45 variableAsToolTip	253
9.116.4.46 variableSubstitutions	253
9.116.4.47 visible	253
9.116.4.48 writeOnClick	253
9.116.4.49 writeOnPress	254
9.116.4.50 writeOnRelease	254
9.117 QERecipe Class Reference	254
9.118 QERecordFieldName Class Reference	256
9.119 QERecordSpec Class Reference	257
9.120 QERecordSpecList Class Reference	257
9.121 QEScript Class Reference	257
9.121.1 Member Enumeration Documentation	260
9.121.1.1 UserLevels	260
9.121.2 Property Documentation	260
9.121.2.1 allowDrop	260
9.121.2.2 displayAlarmState	260
9.121.2.3 int	261
9.121.2.4 userLevelEnabled	261
9.121.2.5 userLevelEngineerStyle	261
9.121.2.6 userLevelScientistStyle	261
9.121.2.7 userLevelUserStyle	261
9.121.2.8 userLevelVisibility	262
9.121.2.9 variableAsToolTip	262
9.121.2.10 visible	262
9.122 QEShape Class Reference	262
9.122.1 Detailed Description	266
9.122.2 Member Enumeration Documentation	266
9.122.2.1 animationOptions	266
9.122.2.2 shapeOptions	266

9.122.2.3 UserLevels	267
9.122.3 Constructor & Destructor Documentation	267
9.122.3.1 QEShape	267
9.122.3.2 QEShape	267
9.122.4 Member Function Documentation	267
9.122.4.1 dbValueChanged1	267
9.122.4.2 dbValueChanged2	267
9.122.4.3 dbValueChanged3	267
9.122.4.4 dbValueChanged4	268
9.122.4.5 dbValueChanged5	268
9.122.4.6 dbValueChanged6	268
9.122.5 Property Documentation	268
9.122.5.1 allowDrop	268
9.122.5.2 animation1	268
9.122.5.3 animation2	268
9.122.5.4 animation3	268
9.122.5.5 animation4	269
9.122.5.6 animation5	269
9.122.5.7 animation6	269
9.122.5.8 color1	269
9.122.5.9 color10	269
9.122.5.10 color2	269
9.122.5.11 color3	269
9.122.5.12 color4	269
9.122.5.13 color5	269
9.122.5.14 color6	270
9.122.5.15 color7	270
9.122.5.16 color8	270
9.122.5.17 color9	270
9.122.5.18 displayAlarmState	270
9.122.5.19 int	270
9.122.5.20 offset1	270
9.122.5.21 offset2	271
9.122.5.22 offset3	271

9.122.5.23offset4	271
9.122.5.24offset5	271
9.122.5.25offset6	271
9.122.5.26point1	271
9.122.5.27point10	271
9.122.5.28point2	271
9.122.5.29point3	271
9.122.5.30point4	272
9.122.5.31point5	272
9.122.5.32point6	272
9.122.5.33point7	272
9.122.5.34point8	272
9.122.5.35point9	272
9.122.5.36scale2	272
9.122.5.37scale3	272
9.122.5.38scale4	272
9.122.5.39scale5	273
9.122.5.40scale6	273
9.122.5.41userLevelEnabled	273
9.122.5.42userLevelEngineerStyle	273
9.122.5.43userLevelScientistStyle	273
9.122.5.44userLevelUserStyle	273
9.122.5.45userLevelVisibility	274
9.122.5.46variable1	274
9.122.5.47variable2	274
9.122.5.48variable3	274
9.122.5.49variable4	274
9.122.5.50variable5	274
9.122.5.51variable6	274
9.122.5.52variableAsToolTip	275
9.122.5.53variableSubstitutions	275
9.122.5.54visible	275
9.123QESlider Class Reference	275
9.123.1 Member Enumeration Documentation	277

9.123.1.1 UserLevels	277
9.123.2 Member Function Documentation	277
9.123.2.1 dbValueChanged	277
9.123.3 Member Data Documentation	277
9.123.3.1 writeOnChange	277
9.123.4 Property Documentation	278
9.123.4.1 allowDrop	278
9.123.4.2 displayAlarmState	278
9.123.4.3 int	278
9.123.4.4 subscribe	278
9.123.4.5 userLevelEnabled	278
9.123.4.6 userLevelEngineerStyle	278
9.123.4.7 userLevelScientistStyle	279
9.123.4.8 userLevelUserStyle	279
9.123.4.9 userLevelVisibility	279
9.123.4.10 variable	279
9.123.4.11 variableAsToolTip	279
9.123.4.12 variableSubstitutions	279
9.123.4.13 visible	280
9.124 QESpinBox Class Reference	280
9.124.1 Member Enumeration Documentation	282
9.124.1.1 UserLevels	282
9.124.2 Member Function Documentation	282
9.124.2.1 dbValueChanged	282
9.124.3 Property Documentation	282
9.124.3.1 allowDrop	282
9.124.3.2 displayAlarmState	282
9.124.3.3 int	283
9.124.3.4 subscribe	283
9.124.3.5 userLevelEnabled	283
9.124.3.6 userLevelEngineerStyle	283
9.124.3.7 userLevelScientistStyle	283
9.124.3.8 userLevelUserStyle	284
9.124.3.9 userLevelVisibility	284

9.124.3.10variable	284
9.124.3.11variableAsToolTip	284
9.124.3.12variableSubstitutions	284
9.124.3.13visible	284
9.125QString Class Reference	285
9.126QStringFormatting Class Reference	285
9.126.1 Member Enumeration Documentation	286
9.126.1.1 arrayActions	286
9.126.1.2 formats	287
9.126.1.3 notations	287
9.127QStringFormattingMethods Class Reference	287
9.128QESTripChart Class Reference	288
9.128.1 Member Function Documentation	290
9.128.1.1 restoreConfiguration	290
9.128.1.2 saveConfiguration	290
9.128.2 Property Documentation	291
9.128.2.1 variableSubstitutions	291
9.129QESTripChartAdjustPVDialog Class Reference	291
9.130QESTripChartContextMenu Class Reference	291
9.130.1 Constructor & Destructor Documentation	292
9.130.1.1 QESTripChartContextMenu	292
9.131QESTripChartItem Class Reference	292
9.132QESTripChartNames Class Reference	293
9.133QESTripChartPushButtonSpecifications Struct Reference	294
9.134QESTripChartRangeDialog Class Reference	295
9.135QESTripChartState Class Reference	295
9.136QESTripChartStateList Class Reference	296
9.137QESTripChartStatistics Class Reference	296
9.138QESTripChartTimeDialog Class Reference	296
9.139QESTripChartToolBar Class Reference	297
9.139.1 Detailed Description	297
9.140QESubstitutedLabel Class Reference	298
9.140.1 Member Data Documentation	298
9.140.1.1 labelText	298

9.140.2 Property Documentation	298
9.140.2.1 textSubstitutions	298
9.141 QEToolTip Class Reference	299
9.142 QEWidget Class Reference	300
9.142.1 Detailed Description	302
9.142.2 Member Function Documentation	303
9.142.2.1 activate	303
9.142.2.2 deactivate	303
9.142.2.3 defaultFileLocation	303
9.142.2.4 doAction	303
9.142.2.5 findQEFile	304
9.142.2.6 findQEFFile	304
9.142.2.7 getColor	304
9.142.2.8 getFrameworkVersion	304
9.142.2.9 getMessageSourceId	304
9.142.2.10getQcalItem	304
9.142.2.11getQWidget	304
9.142.2.12openQEFile	305
9.142.2.13processAlarmInfo	305
9.142.2.14readNow	305
9.142.2.15restoreConfiguration	305
9.142.2.16saveConfiguration	305
9.142.2.17scaleBy	305
9.142.2.18setMessageSourceId	306
9.142.2.19setVariableNameAndSubstitutions	306
9.142.2.20writeNow	306
9.143 QEWidgets Class Reference	306
9.144 qcastatemachine::ReadQCStateMachine Class Reference	306
9.145 recording Class Reference	307
9.146 imageDisplayProperties::rgbPixel Struct Reference	307
9.147 SaveRestoreSignal Class Reference	308
9.147.1 Member Function Documentation	308
9.147.1.1 restore	308
9.147.1.2 save	308

9.148selectMenu Class Reference	308
9.149signalSlotHandler Class Reference	309
9.150standardProperties Class Reference	310
9.151StateMachineTemplate Class Reference	311
9.152qcastatemachine::SubscriptionQCaStateMachine Class Reference	311
9.153trace Class Reference	312
9.154userInfoStruct Class Reference	312
9.155QEPeriodic::userInfoStructArray Struct Reference	313
9.156userLevelSignal Class Reference	313
9.157userLevelSlot Class Reference	313
9.158userLevelTypes Class Reference	314
9.158.1 Member Enumeration Documentation	314
9.158.1.1 userLevels	314
9.159UserMessage Class Reference	314
9.159.1 Detailed Description	316
9.160UserMessageSignal Class Reference	317
9.160.1 Detailed Description	318
9.161UserMessageSlot Class Reference	318
9.161.1 Detailed Description	318
9.162ValueScaling Class Reference	319
9.163VideoWidget Class Reference	319
9.164WidgetRef Class Reference	320
9.165qcastatemachine::WriteQCaStateMachine Class Reference	321
9.166zoomMenu Class Reference	321

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

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

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

See the GNU General Public License for more details.

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

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

Chapter 3

ASgui screen shots

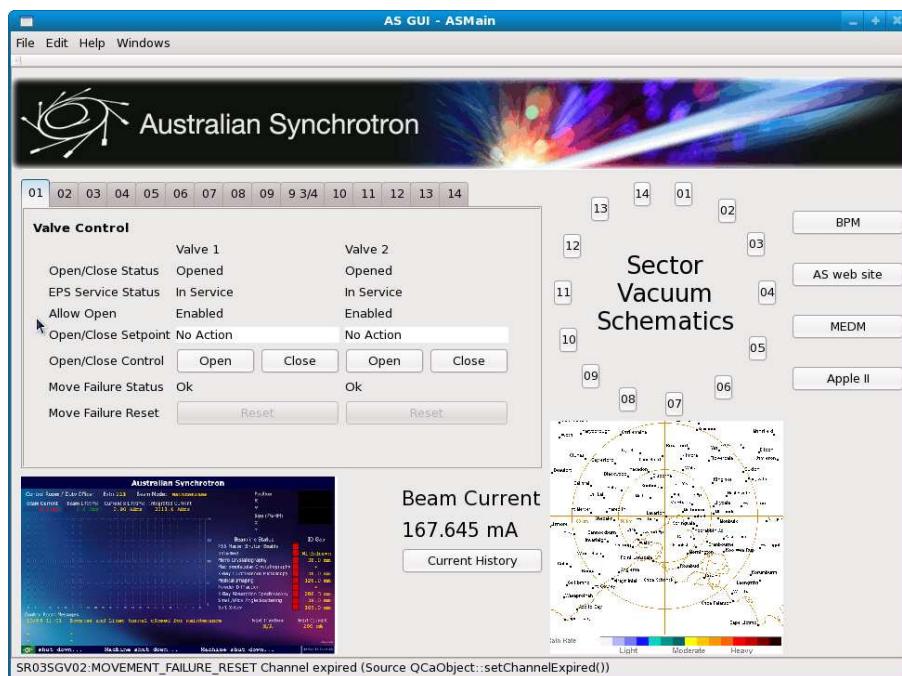


Figure 3.1: Australian Synchrotron mock up

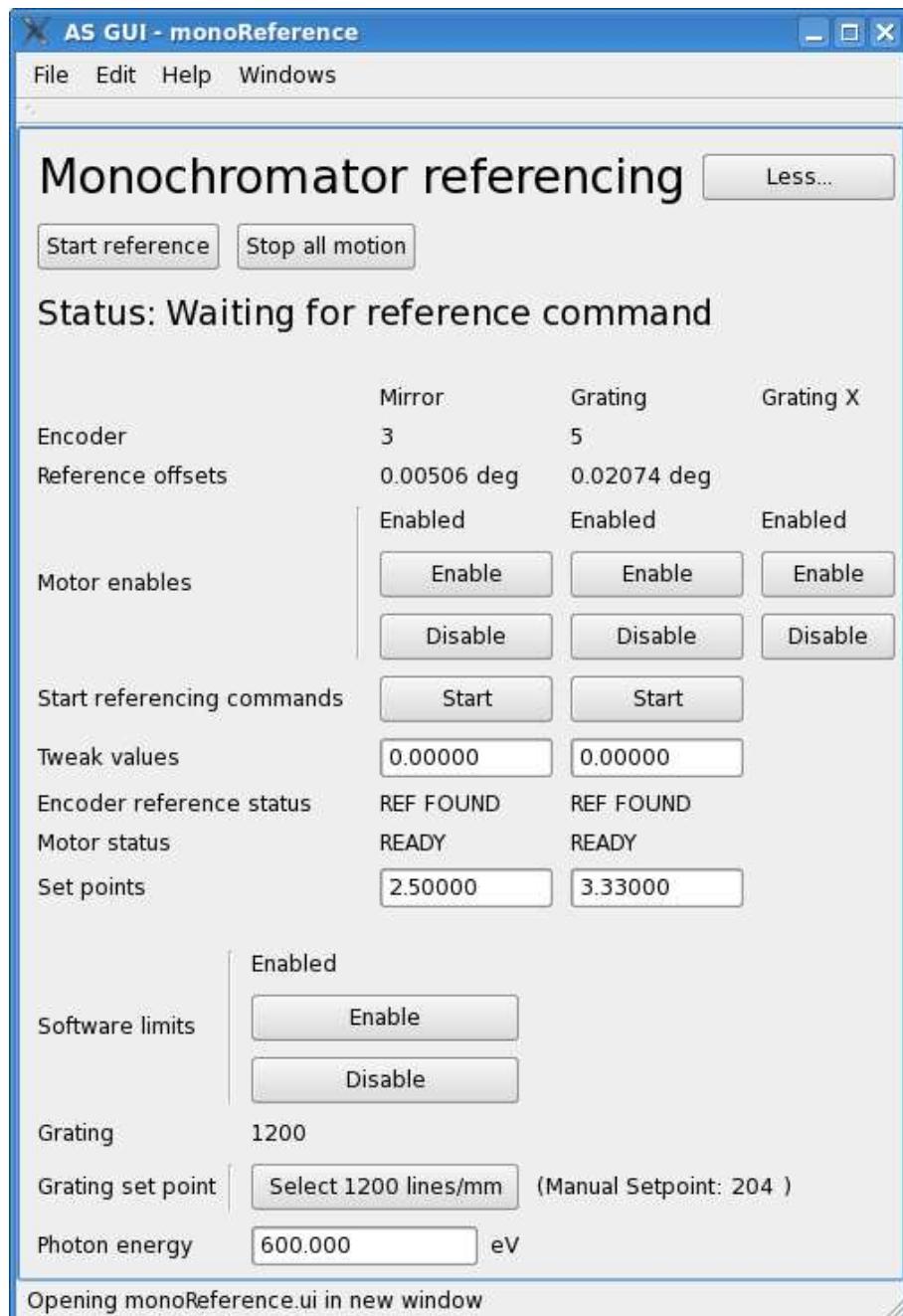


Figure 3.2: Monochromator referencing

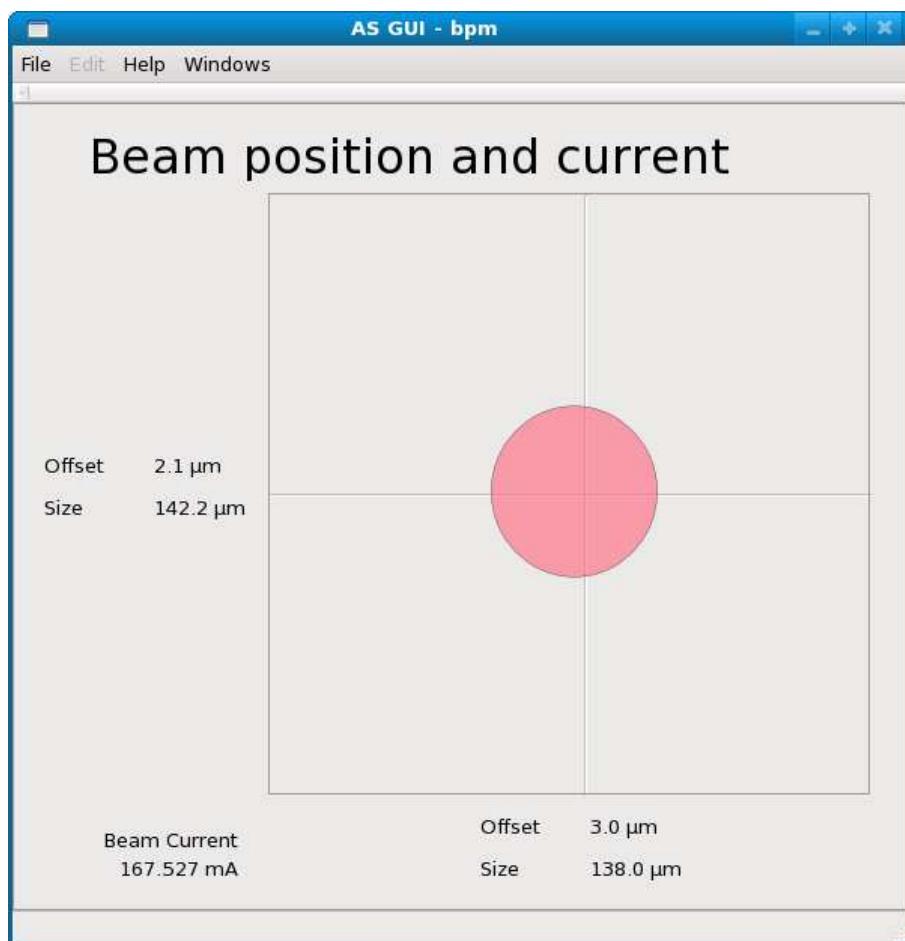


Figure 3.3: Beam position monitor

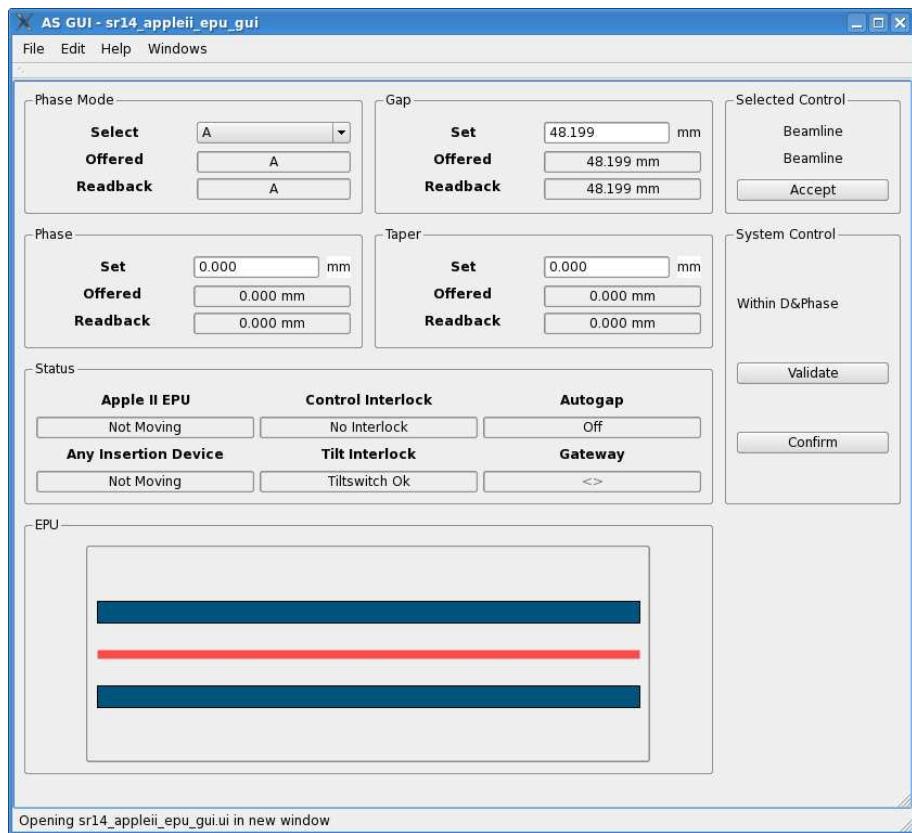


Figure 3.4: Insertion device



Figure 3.5: Injection efficiency monitor

Chapter 4

other applications using epicsqt widgets

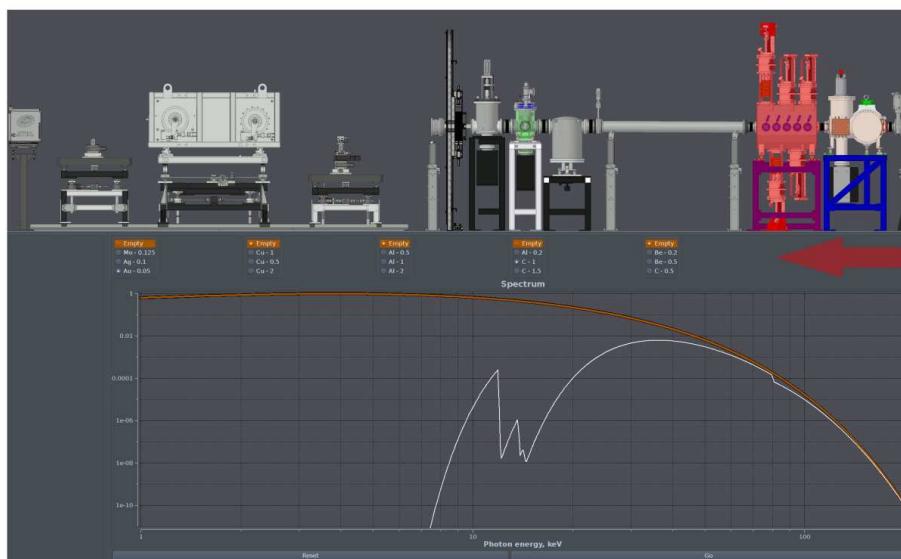


Figure 4.1: Medical Imaging beamline

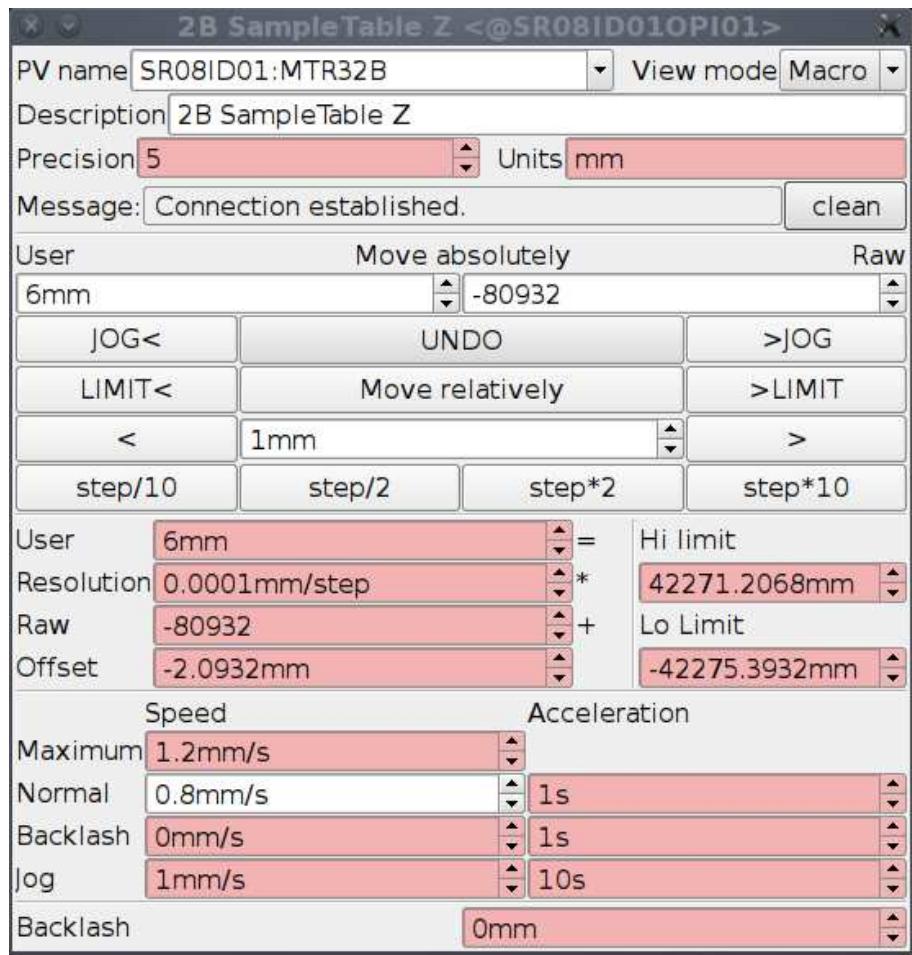


Figure 4.2: Motor controller

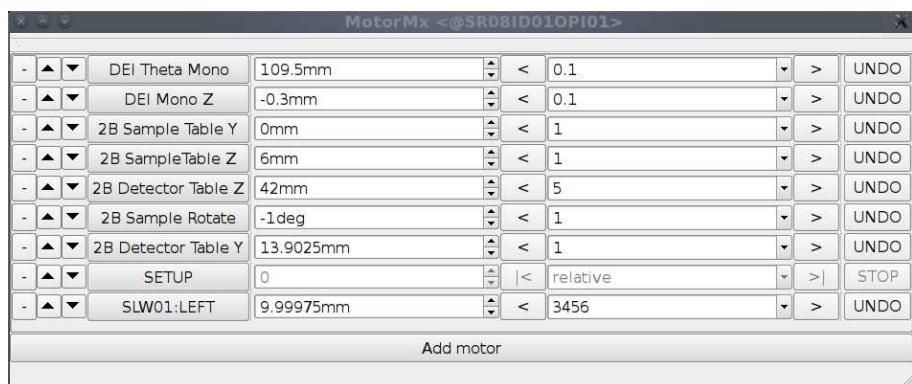


Figure 4.3: Motor controller

Chapter 5

Qt Designer

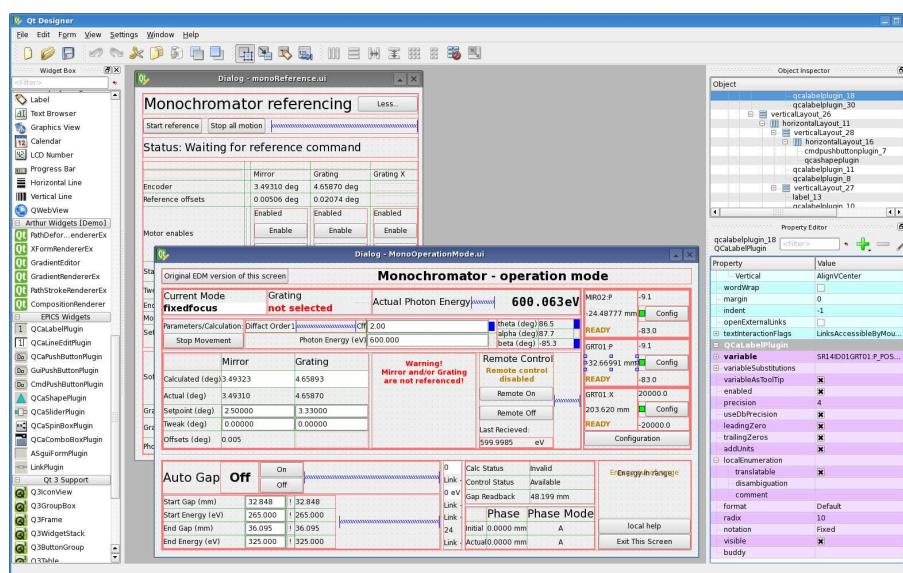


Figure 5.1: Editing multiple GUIs

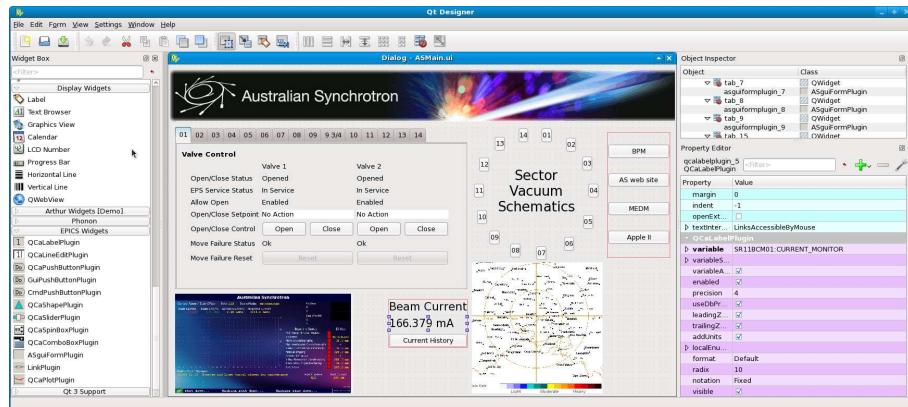


Figure 5.2: Editing a GUI

Chapter 6

Qt Creator

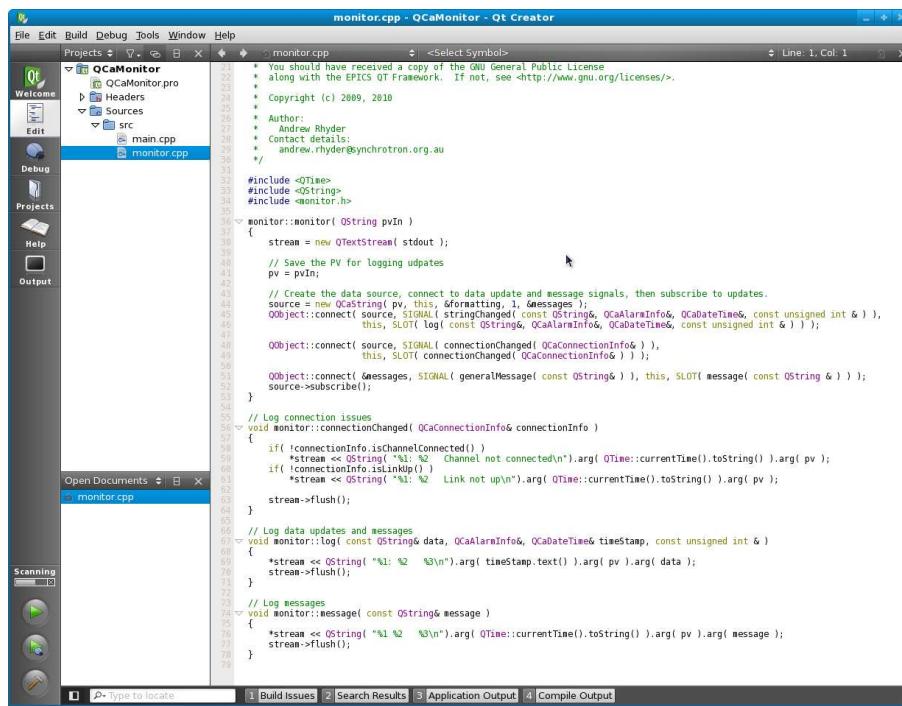


Figure 6.1: Application using epicsqt data source classes

Chapter 7

Class Index

7.1 Class Hierarchy

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

_CopyPaste	29
_Field	29
_Item	30
_QDialogItem	31
_QPushButtonGroup	31
_QTableWidgetFileBrowser	31
_QTableWidgetLog	32
_QTableWidgetScript	32
applicationLauncher	32
areaInfo	33
QEAnalogIndicator::Band	34
QEAnalogIndicator::BandList	34
ContainerProfile	35
QEWidget	300
QEAnalogProgressBar	79
QEBitStatus	88
QEComboBox	108
QEConfiguredLayout	113
QEFileBrowser	117
QEForm	124
QEFrame	126
QE PvProperties	237
QE StripChart	288
QEGenericButton	129
QE CheckBox	93
QE PushButton	221
QE Radio Button	239
QEGenericEdit	131
QELineEdit	187

QENumericEdit	203
QEGroupBox	138
QEImage	142
QELabel	177
QELink	193
QELog	197
QELogin	202
QEPeriodic	207
QEPlot	215
QERecipe	254
QEScript	257
QEShape	262
QESlider	275
QESpinBox	280
QEStripChartItem	292
QESubstitutedLabel	298
contextMenu	36
QEWidget	300
contextMenuObject	38
QEPeriodic::elementInfoStruct	39
FFBuffer	39
FFThread	40
flipRotateMenu	40
fullScreenWindow	40
histogram	41
histogramScroll	41
historicImage	41
imageContextMenu	42
imageDisplayProperties	43
imageInfo	44
QEImage	142
imageMarkup	45
VideoWidget	319
imageUpdateIndicator	47
loginWidget	47
managePixmaps	48
QEGenericButton	129
QELabel	177
markupDisplayMenu	49
markupItem	51
markupBeam	48
markupEllipse	49
markupHLine	50
markupLine	53
markupRegion	54
markupTarget	55
markupText	55
markupVLine	56

message_types	57
mpegSource	57
QEImage	142
mpegSourceObject	58
QEStripChartToolBar::OwnWidgets	59
PeriodicDialog	59
PeriodicElementSetupForm	60
PeriodicSetupDialog	60
PersistanceManager	60
playbackTimer	61
PMContext	61
PMElement	62
PMElementList	62
pointInfo	63
processManager	63
profilePlot	64
PublishedProfile	64
QBitStatus	65
QEBitStatus	88
QCaAlarmInfo	67
QCaConnectionInfo	67
QCaDataPoint	68
QCaDataPointList	68
QCaDateTime	69
QCaEventFilter	69
QCaEventItem	70
QCaEventUpdate	70
QCaInstalledFiltersListItem	70
qcaobject::QCaObject	71
QEByteArray	92
QEFloating	122
QEInteger	174
QString	285
QCaVariableNamePropertyManager	73
QEAnalogIndicator	74
QEAnalogProgressBar	79
QECheckBoxManager	108
QEConfiguredLayoutManager	115
QEDragDrop	116
QWidget	300
QEFloatingArray	123
QEFloatingFormatting	124
QEImageMarkupThickness	174
QEImageOptionsDialog	174
QEIntegerArray	175
QEIntegerFormatting	176
QELineEditManager	192
QELocalEnumeration	195

QELoginDialog	202
QENumericEditManager	206
QEPeriodicComponentData	213
QEPeriodicTaskMenu	213
QEPeriodicTaskMenuFactory	214
QEpicsPV	214
QEPVNameLists	236
QEPvPropertiesManager	239
QERecordFieldName	256
QERecordSpec	257
QERecordSpecList	257
QEStringFormatting	285
QEStringFormattingMethods	287
QEAnalogProgressBar	79
QEGenericButton	129
QELabel	177
QLineEdit	187
QEStripChartAdjustPVDialog	291
QEStripChartContextMenu	291
QEStripChartNames	293
QEStripChartPushButtonSpecifications	294
QEStripChartRangeDialog	295
QEStripChartState	295
QEStripChartStateList	296
QEStripChartStatistics	296
QEStripChartTimeDialog	296
QEStripChartToolBar	297
QEToolTip	299
QEWidget	300
QEWidgets	306
recording	307
imageDisplayProperties::rgbPixel	307
SaveRestoreSignal	308
selectMenu	308
signalSlotHandler	309
standardProperties	310
QEWidget	300
StateMachineTemplate	311
qcastatemachine::QCaStateMachine	73
qcastatemachine::ConnectionQCaStateMachine	34
qcastatemachine::ReadQCaStateMachine	306
qcastatemachine::SubscriptionQCaStateMachine	311
qcastatemachine::WriteQCaStateMachine	321
trace	312
userInfoStruct	312
QEPeriodic::userInfoStructArray	313
userLevelSignal	313
userLevelSlot	313

userLevelTypes	314
UserMessage	314
QEWidget	300
UserMessageSignal	317
UserMessageSlot	318
ValueScaling	319
WidgetRef	320
zoomMenu	321

Chapter 8

Class Index

8.1 Class List

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

_CopyPaste	29
_Field	29
_Item	30
_QDialogItem	31
_QPushButtonGroup	31
_QTableWidgetFileBrowser	31
_QTableWidgetLog	32
_QTableWidgetScript	32
applicationLauncher	32
areaInfo	33
QEAnalogIndicator::Band	34
QEAnalogIndicator::BandList	34
qcastatemachine::ConnectionQCaStateMachine	34
ContainerProfile	35
contextMenu	36
contextMenuObject	38
QEPeriodic::elementInfoStruct	39
FFBuffer	39
FFThread	40
flipRotateMenu	40
fullScreenWindow	40
histogram	41
histogramScroll	41
historicImage	41
imageContextMenu	42
imageDisplayProperties	43
imageInfo	44
imageMarkup	45
imageUpdateIndicator	47

loginWidget	47
managePixmaps	48
markupBeam	48
markupDisplayMenu	49
markupEllipse	49
markupHLine	50
markupItem	51
markupLine	53
markupRegion	54
markupTarget	55
markupText	55
markupVLine	56
message_types	57
mpegSource	57
mpegSourceObject	58
QEStripChartToolBar::OwnWidgets	59
PeriodicDialog	59
PeriodicElementSetupForm	60
PeriodicSetupDialog	60
PersistanceManager	60
playbackTimer	61
PMContext	61
PMElement	62
PMElementList	62
pointInfo	63
processManager	63
profilePlot	64
PublishedProfile	64
QBitStatus	65
QCaAlarmInfo	67
QCaConnectionInfo	67
QCaDataPoint	68
QCaDataPointList	68
QCaDateTime	69
QCaEventFilter	69
QCaEventItem	70
QCaEventUpdate	70
QCaInstalledFiltersListItem	70
qcaobject::QCaObject	71
qcastatemachine::QCaStateMachine	73
QCaVariableNamePropertyManager	73
QEAnalogIndicator	74
QEAnalogProgressBar	79
QEBitStatus	88
QEByteArray	92
QECheckBox	93
QECheckBoxManager	108
QEComboBox	108
QEConfiguredLayout	113
QEConfiguredLayoutManager	115

QEDragDrop	116
QEFileBrowser	117
QEFloating	122
QEFloatingArray	123
QEFloatingFormatting	124
QEForm	124
QEFrame	126
QEGenericButton	129
QEGenericEdit	131
QEGroupBox	138
QEImage	142
QEImageMarkupThickness	174
QEImageOptionsDialog	174
QEInteger	174
QEIntegerArray	175
QEIntegerFormatting	176
QELabel	177
QELineEdit	187
QELineEditManager	192
QELink	193
QELocalEnumeration	195
QELog	197
QELogin	202
QELoginDialog	202
QENumericEdit (The QENumericEdit class This class is similar to QELineEdit (both of which are derived from QLineEdit). However this class is tailored specifically for editing numerical values)	203
QENumericEditManager	206
QEPeriodic	207
QEPeriodicComponentData	213
QEPeriodicTaskMenu	213
QEPeriodicTaskMenuFactory	214
QEpicsPV	214
QEPlot	215
QEPushButton	221
QEPVNameLists	236
QEPvProperties	237
QEPvPropertiesManager	239
QERadioButton	239
QERecipe	254
QERecordFieldName	256
QERecordSpec	257
QERecordSpecList	257
QEScript	257
QEShape	262
QESlider	275
QESpinBox	280
QEString	285
QEStringFormatting	285
QEStringFormattingMethods	287

QEStripChart	288
QEStripChartAdjustPVDialo	291
QEStripChartContextMenu	291
QEStripChartItem	292
QEStripChartNames	293
QEStripChartPushButtonSpecifi	294
QEStripChartRangeDialog	295
QEStripChartState	295
QEStripChartStateList	296
QEStripChartStatistics	296
QEStripChartTimeDialog	296
QEStripChartToolBar (This class holds all the StripChart tool bar widgets)	297
QESubstitutedLabel	298
QEToolTip	299
QEWidget	300
QEWidgets	306
qcastatemachine::ReadQCaStateMachine	306
recording	307
imageDisplayProperties::rgbPixel	307
SaveRestoreSignal	308
selectMenu	308
signalSlotHandler	309
standardProperties	310
StateMachineTemplate	311
qcastatemachine::SubscriptionQCaStateMachine	311
trace	312
userInfoStruct	312
QEPeriodic::userInfoStructArra	313
userLevelSignal	313
userLevelSlot	313
userLevelTypes	314
UserMessage	314
UserMessageSignal	317
UserMessageSlot	318
ValueScaling	319
VideoWidget	319
WidgetRef	320
qcastatemachine::WriteQCaStateMachine	321
zoomMenu	321

Chapter 9

Class Documentation

9.1 _CopyPaste Class Reference

Public Member Functions

- **_CopyPaste** (bool pEnable, QString pProgram, QString pParameters, int pTimeOut, bool pStop, bool pLog)
- void **setEnable** (bool pEnable)
- bool **getEnable** ()
- void **setProgram** (QString pProgram)
- QString **getProgram** ()
- void **setParameters** (QString pParameters)
- QString **getParameters** ()
- void **setTimeOut** (int pTimeOut)
- int **getTimeOut** ()
- void **setStop** (bool pStop)
- bool **getStop** ()
- void **setLog** (bool pLog)
- bool **getLog** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEScript/QEScript.h
- /tmp/epicsqt/trunk/framework/widgets/QEScript/QEScript.cpp

9.2 _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:

- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.cpp

9.3 [_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:

- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.cpp

9.4 _QDialogItem Class Reference

Public Member Functions

- **_QDialogItem** (QWidget *pParent=0, QString pItemName="", QString pGroupName="", QList<[_Field](#) *> *pCurrentFieldList=0, Qt::WindowFlags pF=0)

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

- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.cpp

9.5 _QPushButtonGroup Class Reference

Public Slots

- void **buttonGroupClicked** ()

Public Member Functions

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

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

- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.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:

- /tmp/epicsqt/trunk/framework/widgets/QEFileBrowser/QEFileBrowser.h
- /tmp/epicsqt/trunk/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:

- /tmp/epicsqt/trunk/framework/widgets/QELog/QELog.h
- /tmp/epicsqt/trunk/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:

- /tmp/epicsqt/trunk/framework/widgets/QEScript/QEScript.h
- /tmp/epicsqt/trunk/framework/widgets/QEScript/QEScript.cpp

9.9 applicationLauncher Class Reference

Public Types

- enum **programStartupOptions** { **PSO_NONE**, **PSO_TERMINAL**, **PSO_LOGOUTPUT**, **PSO_STDOUPUT** }

Public Member Functions

- void **launchImage** (VariableNameManager *variableNameManager, QImage image)
- void **launch** (VariableNameManager *variableNameManager, QObject *receiver)
- void **launchCommon** (VariableNameManager *variableNameManager, QTemporaryFile *tempFile=NULL, QObject *receiver=NULL)

- void **setProgram** (QString programIn)
- QString **getProgram** ()
- void **setArguments** (QStringList argumentsIn)
- QStringList **getArguments** ()
- void **setProgramStartupOption** (programStartupOptions programStartupOptionIn)

- programStartupOptions **getProgramStartupOption** ()

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

- /tmp/epicsqt/trunk/framework/widgets/include/applicationLauncher.h
- /tmp/epicsqt/trunk/framework/widgets/src/applicationLauncher.cpp

9.10 arealInfo Class Reference

Public Member Functions

- void **setX1** (long x)
- void **setY1** (long y)
- void **setX2** (long x)
- void **setY2** (long y)
- void **setX** (long x)
- void **setY** (long y)
- void **setW** (long w)
- void **setH** (long h)
- void **setPoint1** (QPoint p1In)
- void **setPoint2** (QPoint p2In)
- void **clearX1** ()
- void **clearY1** ()
- void **clearX2** ()
- void **clearY2** ()
- void **clearX** ()
- void **clearY** ()
- void **clearW** ()
- void **clearH** ()
- bool **getStatus** ()
- QRect **getArea** ()
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/QEImage.h

9.11 QEAnalogIndicator::Band Struct Reference

Public Attributes

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

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

- /tmp/epicsqt/trunk/framework/widgets/QEAnalogIndicator/QEAnalogIndicator.h

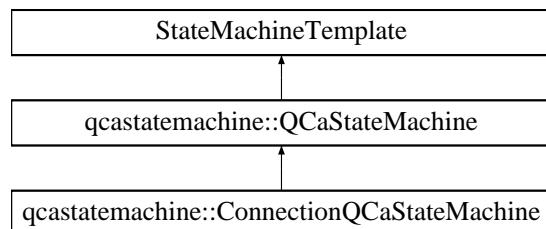
9.12 QEAnalogIndicator::BandList Class Reference

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

- /tmp/epicsqt/trunk/framework/widgets/QEAnalogIndicator/QEAnalogIndicator.h

9.13 qcastatemachine::ConnectionQCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::ConnectionQCaStateMachine:



Public Member Functions

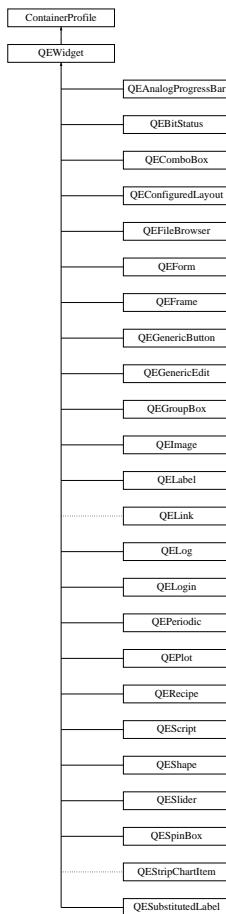
- **ConnectionQCaStateMachine** (void *parent)
- bool **process** (int requestedState)

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

- /tmp/epicsqt/trunk/framework/data/include/QCaStateMachine.h
- /tmp/epicsqt/trunk/framework/data/src/QCaStateMachine.cpp

9.14 ContainerProfile Class Reference

Inheritance diagram for ContainerProfile:



Public Member Functions

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

- void **addMacroSubstitutions** (QString macroSubstitutionsIn)
- void **removeMacroSubstitutions** ()
- void **addPriorityMacroSubstitutions** (QString macroSubstitutionsIn)

- void **removePriorityMacroSubstitutions** ()
- QObject * **getGuiLaunchConsumer** ()
- QString **getPath** ()
- QStringList **getPathList** ()
- QString **getParentPath** ()
- void **setPublishedParentPath** (QString publishedParentPathIn)
- QString **getMacroSubstitutions** ()
- bool **isProfileDefined** ()
- bool **areUserLevelPasswordsSet** ()
- QStringList **getEnvPathList** ()
- QString **getUserLevelPassword** ([userLevelTypes::userLevels](#) level)
- void **setUserLevelPassword** ([userLevelTypes::userLevels](#) level, QString passwordIn)
- void **addContainedWidget** ([QEWidget](#) *containedWidget)
- [QEWidget](#) * **getNextContainedWidget** ()
- void **removeContainedWidget** ([QEWidget](#) *containedWidget)
- unsigned int **getMessageFormId** ()
- unsigned int **getPublishedMessageFormId** ()
- void **setPublishedMessageFormId** (unsigned int publishedMessageFormIdIn)
- bool **setDontActivateYet** (bool dontActivateIn)
- bool **getDontActivateYet** ()
- void **releaseProfile** ()
- void **publishOwnProfile** ()
- void **setUserLevel** ([userLevelTypes::userLevels](#) level)
- [userLevelTypes::userLevels](#) **getUserLevel** ()
- virtual void **userLevelChangedGeneral** ([userLevelTypes::userLevels](#))
- [PersistanceManager](#) * **getPersistanceManager** ()

Static Public Member Functions

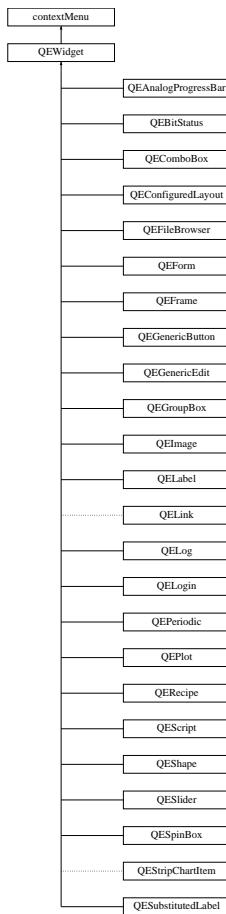
- static QChar **platformSeperator** ()

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

- /tmp/epicsqt/trunk/framework/widgets/include/ContainerProfile.h
- /tmp/epicsqt/trunk/framework/widgets/src/ContainerProfile.cpp

9.15 contextMenu Class Reference

Inheritance diagram for contextMenu:



Public Types

- enum **contextMenuOptions** {

 CM_NOOPTION, CM_COPY_VARIABLE, CM_COPY_DATA, CM_PASTE,

 CM_DRAG_VARIABLE, CM_DRAG_DATA, CM_SHOW_PV_PROPERTIES, CM_-

 ADD_TO_STRIPCHART,

 CM_ADD_TO_SCRATCH_PAD, CM_GENERAL_PV_EDIT, CM_SPECIFIC_WIDGETS_-

 START_HERE }

Public Member Functions

- **contextMenu** ([QEWidget](#) *qewIn)
- void **setConsumer** ([QObject](#) *consumer)
- void **setupContextMenu** ()
- bool **isDraggingVariable** ()
- [QMenu](#) * **buildContextMenu** ()

- void **contextMenuTriggered** (int selectedItemNum)
- virtual QString **copyVariable** ()
- virtual QVariant **copyData** ()
- virtual void **paste** (QVariant)
- QAction * **showContextMenuGlobal** (const QPoint &globalPos)
- QAction * **showContextMenu** (const QPoint &pos)
- QAction * **showContextMenuGlobal** (QMenu *menu, const QPoint &globalPos)

- QAction * **showContextMenu** (QMenu *menu, const QPoint &pos)
- void **addMenuItem** (QMenu *menu, const QString &title, const bool checkable, const bool checked, const int option)

Friends

- class [contextMenuObject](#)

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

- /tmp/epicsqt/trunk/framework/widgets/include/contextMenu.h
- /tmp/epicsqt/trunk/framework/widgets/src/contextMenu.cpp

9.16 contextMenuObject Class Reference

Public Slots

- void **contextMenuTriggeredSlot** (QAction *selectedItem)
- void **showContextMenuSlot** (const QPoint &pos)

Signals

- void **requestAction** (const QEActionRequests &)

Public Member Functions

- **contextMenuObject** ([contextMenu](#) *menuIn)
- void **sendRequestAction** (const QEActionRequests &request)

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

- /tmp/epicsqt/trunk/framework/widgets/include/contextMenu.h
- /tmp/epicsqt/trunk/framework/widgets/src/contextMenu.cpp

9.17 QEPeriodic::elementInfoStruct Struct Reference

Public Attributes

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

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodic.h

9.18 FFBuffer Class Reference

Public Member Functions

- void **reserve** ()
- void **release** ()
- bool **grabFree** ()

Public Attributes

- QMutex * **mutex**
- unsigned char * **mem**
- AVFrame * **pFrame**
- PixelFormat **pix_fmt**
- int **width**
- int **height**
- int **refs**

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/mpeg.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/mpeg.cpp

9.19 FFThread Class Reference

Public Slots

- void **stopGracefully ()**

Signals

- void **updateSignal (FFBuffer *buf)**

Public Member Functions

- **FFThread** (const QString &url, QObject *parent)
- void **run ()**

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/mpeg.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/mpeg.cpp

9.20 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:

- /tmp/epicsqt/trunk/framework/widgets/QElImage/flipRotateMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/flipRotateMenu.cpp

9.21 fullScreenWindow Class Reference

Signals

- void **fullScreenResize ()**

Public Member Functions

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

Protected Member Functions

- void **resizeEvent** (QResizeEvent *event)

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/fullScreenWindow.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/fullScreenWindow.cpp

9.22 histogram Class Reference

Public Member Functions

- **histogram** (QWidget *parent, [imageDisplayProperties](#) *idp)

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/brightnessContrast.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/brightnessContrast.cpp

9.23 histogramScroll Class Reference

Public Member Functions

- **histogramScroll** (QWidget *parent, [imageDisplayProperties](#) *idp)

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/brightnessContrast.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/brightnessContrast.cpp

9.24 historicImage Class Reference

Public Member Functions

- **historicImage** (QByteArray image, unsigned long dataSize, [QCaAlarmInfo](#) &alarmInfo, [QCaDateTime](#) &time)

Public Attributes

- QByteArray **image**
- unsigned long **dataSize**

- [QCaAlarmInfo](#) `alarmInfo`
- [QCaDateTime](#) `time`

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

- [/tmp/epicsqt/trunk/framework/widgets/QElImage/recording.h](#)
- [/tmp/epicsqt/trunk/framework/widgets/QElImage/QElImage.cpp](#)

9.25 imageContextMenu Class Reference

Public Types

- enum `imageContextMenuOptions` {
 ICM_NONE = contextMenu::CM_SPECIFIC_WIDGETS_START_HERE, **ICM_SAVE**,
 ICM_PAUSE, **ICM_ENABLE_TIME**,
 ICM_ENABLE_CURSOR_PIXEL, **ICM_ABOUT_IMAGE**, **ICM_ENABLE_VERT**,
 ICM_ENABLE_HOZ,
 ICM_ENABLE_AREA1, **ICM_ENABLE_AREA2**, **ICM_ENABLE_AREA3**, **ICM_ENABLE_AREA4**,
 ICM_ENABLE_LINE, **ICM_ENABLE_TARGET**, **ICM_ENABLE_BEAM**, **ICM_DISPLAY_BUTTON_BAR**,
 ICM_DISPLAY_IMAGE_DISPLAY_PROPERTIES, **ICM_DISPLAY_RECORDER**,
 ICM_ZOOM_SELECTED, **ICM_ZOOM_FIT**,
 ICM_ZOOM_PLUS, **ICM_ZOOM_MINUS**, **ICM_ZOOM_10**, **ICM_ZOOM_25**,
 ICM_ZOOM_50, **ICM_ZOOM_75**, **ICM_ZOOM_100**, **ICM_ZOOM_150**,
 ICM_ZOOM_200, **ICM_ZOOM_300**, **ICM_ZOOM_400**, **ICM_ROTATE_NONE**,
 ICM_ROTATE_RIGHT, **ICM_ROTATE_LEFT**, **ICM_ROTATE_180**, **ICM_FLIP_HORIZONTAL**,
 ICM_FLIP_VERTICAL, **ICM_SELECT_PAN**, **ICM_SELECT_HSLICE**, **ICM_SELECT_VSLICE**,
 ICM_SELECT_AREA1, **ICM_SELECT_AREA2**, **ICM_SELECT_AREA3**, **ICM_SELECT_AREA4**,
 ICM_SELECT_PROFILE, **ICM_SELECT_TARGET**, **ICM_SELECT_BEAM**, **ICM_CLEAR_MARKUP**,
 ICM_THICKNESS_ONE_MARKUP, **ICM_THICKNESS_SELECT_MARKUP**, **ICM_COPY_PLOT_DATA**, **ICM_FULL_SCREEN**,
 ICM_DISPLAY_HSLICE, **ICM_DISPLAY_VSLICE**, **ICM_DISPLAY_AREA1**, **ICM_DISPLAY_AREA2**,
 ICM_DISPLAY_AREA3, **ICM_DISPLAY_AREA4**, **ICM_DISPLAY_PROFILE**, **ICM_DISPLAY_TARGET**,
 ICM_DISPLAY_BEAM, **ICM_DISPLAY_TIMESTAMP**, **ICM_DISPLAY_ELLIPSE**,
 ICM_OPTIONS }

Public Member Functions

- **imageContextMenu** (QWidget *parent=0)
- void **getContextMenuItemOption** (const QPoint &, imageContextMenuOptions *option, bool *checked)
- void **addMenuItem** (const QString &title, const bool checkable, const bool checked, const imageContextMenuOptions option)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageContextMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageContextMenu.cpp

9.26 imageDisplayProperties Class Reference

Classes

- struct [rgbPixel](#)

Signals

- void **brightnessContrastAutoImage** ()
- void **imageDisplayPropertiesChange** ()

Public Member Functions

- void **setBrightnessContrast** (const unsigned int max, const unsigned int min)
- void **setAutoBrightnessContrast** (bool autoBrightnessContrast)
- void **setContrastReversal** (bool contrastReversal)
- void **setLog** (bool log)
- void **setFalseColour** (bool falseColour)
- bool **getAutoBrightnessContrast** ()
- bool **getContrastReversal** ()
- bool **getLog** ()
- bool **getFalseColour** ()
- int **getLowPixel** ()
- int **getHighPixel** ()
- void **setStatistics** (unsigned int minPln, unsigned int maxPln, unsigned int bitDepth, unsigned int binsIn[HISTOGRAM_BINS], [rgbPixel](#) pixelLookup[256])
- void **initialiseImageStats** ()
- void **setHistZoom** (int value)
- int **getHistZoom** ()

Public Attributes

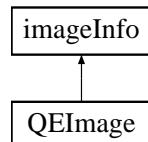
- int **zeroValue**
- int **fullValue**
- bool **defaultFullValue**
- unsigned int **range**
- unsigned int **maxP**
- unsigned int **minP**
- unsigned int **depth**
- unsigned int * **bins**
- **rgbPixel** * **pixelLookup**
- QLabel * **histXLabel**

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/brightnessContrast.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/brightnessContrast.cpp

9.27 imageInfo Class Reference

Inheritance diagram for imageInfo:



Public Member Functions

- void **showInfo** (bool show)
- QLayout * **getInfoWidget** ()
- void **infoShow** (const bool show)
- void **infoUpdateTarget** ()
- void **infoUpdateTarget** (const int x, const int y)
- void **infoUpdateBeam** ()
- void **infoUpdateBeam** (const int x, const int y)
- void **infoUpdateVertProfile** ()
- void **infoUpdateVertProfile** (const int x, const unsigned int thickness)
- void **infoUpdateHozProfile** ()
- void **infoUpdateHozProfile** (const int y, const unsigned int thickness)
- void **infoUpdateProfile** ()
- void **infoUpdateProfile** (const QPoint start, const QPoint end, const unsigned int thickness)

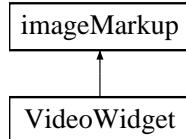
- void **infoUpdateRegion** (const unsigned int region)
- void **infoUpdateRegion** (const unsigned int region, const int x1, const int y1, const int x2, const int y2)
- void **infoUpdatePixel** ()
- void **infoUpdatePixel** (const QPoint pos, int value)
- void **infoUpdateZoom** ()
- void **infoUpdateZoom** (int value)
- void **infoUpdatePaused** ()
- void **infoUpdatePaused** (bool paused)
- void **setBriefInfoArea** (const bool briefIn)
- bool **getBriefInfoArea** ()
- void **freshImage** (QDateTime &time)

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/imageInfo.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/imageInfo.cpp

9.28 imageMarkup Class Reference

Inheritance diagram for imageMarkup:



Public Types

- enum **markupIds** {

 MARKUP_ID_REGION1, **MARKUP_ID_REGION2**, **MARKUP_ID_REGION3**, **MARKUP_ID_REGION4**,

 MARKUP_ID_H_SLICE, **MARKUP_ID_V_SLICE**, **MARKUP_ID_LINE**, **MARKUP_ID_TARGET**,

 MARKUP_ID_BEAM, **MARKUP_ID_TIMESTAMP**, **MARKUP_ID_ELLIPSE**, **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)
- void **markupRegionValueChange** (int areaIndex, QRect area, bool displayMarkups)

- void **markupHProfileChange** (int y, bool displayMarkups)
- void **markupVProfileChange** (int x, bool displayMarkups)
- void **markupLineProfileChange** (QPoint start, QPoint end, bool displayMarkups)

- void **markupTargetValueChange** (QPoint point, bool displayMarkups)
- void **markupBeamValueChange** (QPoint point, bool displayMarkups)
- void **markupEllipseValueChange** (QPoint point1, QPoint point2, bool displayMarkups)
- void **markupValueChange** (int markup, bool displayMarkups, QPoint p1, QPoint p2=QPoint())
- QCursors **getCircleCursor** ()
- QCursors **getTargetCursor** ()
- QCursors **getVLineCursor** ()
- QCursors **getHLineCursor** ()
- QCursors **getLineCursor** ()
- QCursors **getRegionCursor** ()
- virtual void **markupSetCursor** (QCursors cursor)=0
- void **setMarkupLegend** (markupIds mode, QString legend)
- QString **getMarkupLegend** (markupIds mode)
- void **clearMarkup** (markupIds markupId)
- void **showMarkup** (markupIds markupId)
- void **displayMarkup** (markupIds markupId, bool state)
- bool **isMarkupVisible** (markupIds mode)

Public Attributes

- QVector< **markupItem** * > **items**
- QPoint **grabOffset**
- bool **markupAreasStale**
- QFont **legendFont**
- QFontMetrics * **legendFontMetrics**

Protected Member Functions

- void **drawMarkups** (QPainter &p, const QRect &rect)
- bool **anyVisibleMarkups** ()
- QCursors **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** (const QSize &newSize, const QSize &oldSize, const double scale)
- virtual void **markupChange** (QVector< QRect > &changedAreas)=0
- virtual void **markupAction** (markupIds mode, bool complete, bool clearing, QPoint point1, QPoint point2, unsigned int thickness)=0

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

9.29 imageUpdateIndicator Class Reference

Public Member Functions

- void **freshImage** ()
- void **paintEvent** (QPaintEvent *)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageInfo.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageInfo.cpp

9.30 loginWidget Class Reference

Public Member Functions

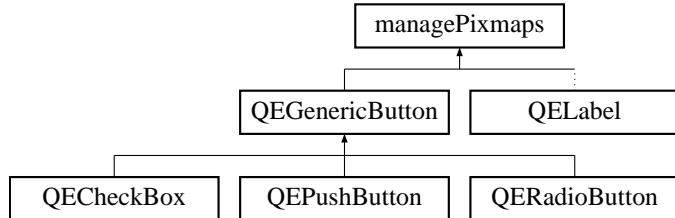
- **loginWidget** (QELogin *ownerIn)
- **userLevelTypes::userLevels getUserType** ()
- QString **getPassword** ()
- void **clearPassword** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.h
- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.cpp

9.31 managePixmaps Class Reference

Inheritance diagram for managePixmaps:



Public Member Functions

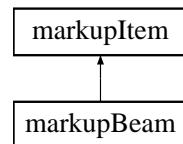
- void **setDataPixmap** (const QPixmap &Pixmap, const unsigned int index)
- QPixmap **getDataPixmap** (const unsigned int index)
- QPixmap **getDataPixmap** (const QString value)

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

- /tmp/epicsqt/trunk/framework/widgets/include/managePixmaps.h
- /tmp/epicsqt/trunk/framework/widgets/src/managePixmaps.cpp

9.32 markupBeam Class Reference

Inheritance diagram for markupBeam:



Public Member Functions

- **markupBeam** (*imageMarkup* *ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor *cursor)
- QPoint **origin** ()

- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupBeam.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupBeam.cpp

9.33 markupDisplayMenu Class Reference

Public Member Functions

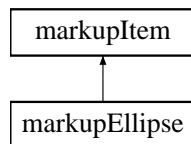
- **markupDisplayMenu** (QWidget *parent=0)
- void **setDisplayed** (imageContextMenu::imageContextMenuOptions option, bool state)
- bool **isDisplayed** (imageContextMenu::imageContextMenuOptions option)
- void **enable** (imageContextMenu::imageContextMenuOptions option, bool state)

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupDisplayMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupDisplayMenu.cpp

9.34 markupEllipse Class Reference

Inheritance diagram for markupEllipse:



Public Member Functions

- **markupEllipse** ([imageMarkup](#) *ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)

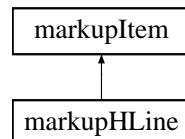
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor *cursor)
- QPoint **origin** ()
- QCursors **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursors **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupEllipse.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupEllipse.cpp

9.35 markupHLine Class Reference

Inheritance diagram for markupHLine:



Public Member Functions

- **markupHLine** (imageMarkup *ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursors *cursor)
- QPoint **origin** ()
- QCursors **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursors **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

9.35.1 Member Function Documentation

9.35.1.1 void markupHLine::drawMarkup (QPainter & p) [virtual]

!! draw the handle in the middle of the existing view, not the entire image

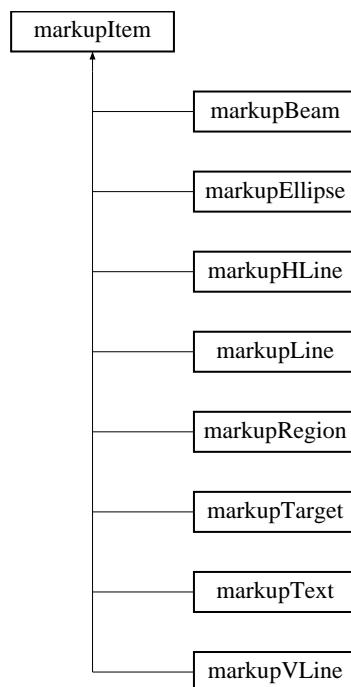
Implements [markupItem](#).

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupHLine.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupHLine.cpp

9.36 markupItem Class Reference

Inheritance diagram for markupItem:



Public Types

- enum **markupHandles** {
 MARKUP_HANDLE_NONE, **MARKUP_HANDLE_START**, **MARKUP_HANDLE_END**, **MARKUP_HANDLE_CENTER**,
 MARKUP_HANDLE_TL, **MARKUP_HANDLE_TR**, **MARKUP_HANDLE_BL**, **MARKUP_HANDLE_BR**,

MARKUP_HANDLE_T, MARKUP_HANDLE_B, MARKUP_HANDLE_L, MARKUP_HANDLE_R }

Public Member Functions

- void **drawMarkupItem** (QPainter &p)
- void **setColor** (QColor colorIn)
- void **scale** (const double xScale, const double yScale, const double zoomScale)
- void **setImageSize** (const QSize &newSize)
- virtual QPoint **origin** ()=0
- virtual void **moveTo** (const QPoint pos)=0
- virtual void **startDrawing** (const QPoint pos)=0
- virtual bool **isOver** (const QPoint point, QCursor *cursor)=0
- virtual QCursor **cursorForHandle** (const markupItem::markupHandles handle)=0

- virtual QPoint **getPoint1** ()=0
- virtual QPoint **getPoint2** ()=0
- virtual QCursor **defaultCursor** ()=0
- virtual void **nonInteractiveUpdate** (QPoint, QPoint)
- void **setThickness** (const unsigned int thicknessIn)
- unsigned int **getThickness** ()
- void **setLegend** (const QString legendIn)
- const QString **getLegend** ()

Public Attributes

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

Protected Types

- enum **isOverOptions** { **OVER_LINE**, **OVER_BORDER**, **OVER_AREA** }
- enum **legendJustification** { **ABOVE_RIGHT**, **BELOW_LEFT**, **BELOW_RIGHT** }

Protected Member Functions

- **markupItem** ([imageMarkup](#) *ownerIn, const isOverOptions over, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- virtual void **setArea** ()=0
- virtual void **drawMarkup** (QPainter &p)=0
- bool **pointIsNear** (QPoint p1, QPoint p)
- QColor **getColor** ()

- const QSize **getLegendSize** ()
- void **addLegendArea** ()
- const QPoint **setLegendPos** (QPoint pos, legendJustification just)
- const QPoint **getLegendPos** ()
- void **drawLegend** (QPainter &p, QPoint pos, legendJustification just)
- QPoint **limitPointToImage** (const QPoint pos)

Protected Attributes

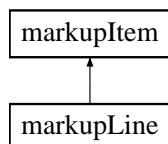
- markupHandles **activeHandle**
- **imageMarkup** * **owner**
- QSize **imageSize**
- unsigned int **thickness**
- unsigned int **maxThickness**

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupItem.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupItem.cpp

9.37 markupLine Class Reference

Inheritance diagram for markupLine:



Public Member Functions

- **markupLine** (**imageMarkup** *ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor *cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()

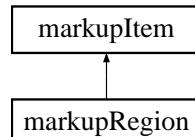
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupLine.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupLine.cpp

9.38 markupRegion Class Reference

Inheritance diagram for markupRegion:



Public Member Functions

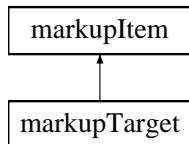
- **markupRegion** ([imageMarkup](#) *ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursors *cursor)
- QPoint **origin** ()
- QCursors **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursors **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupRegion.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupRegion.cpp

9.39 markupTarget Class Reference

Inheritance diagram for markupTarget:



Public Member Functions

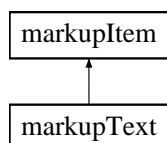
- **markupTarget** ([imageMarkup](#) *ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor *cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/markupTarget.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/markupTarget.cpp

9.40 markupText Class Reference

Inheritance diagram for markupText:



Public Member Functions

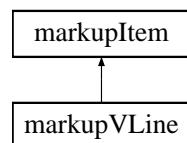
- **markupText** (*imageMarkup* *ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **setText** (QString textIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor *cursor)
- QPoint **origin** ()
- QCursors **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- QCursors **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupText.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupText.cpp

9.41 markupVLine Class Reference

Inheritance diagram for markupVLine:



Public Member Functions

- **markupVLine** (*imageMarkup* *ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursors *cursor)
- QPoint **origin** ()
- QCursors **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()

- QPoint **getPoint2** ()
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)
- void **nonInteractiveUpdate** (QPoint p1, QPoint p2)

9.41.1 Member Function Documentation

9.41.1.1 void markupVLine::drawMarkup (QPainter & p) [virtual]

!! draw the handle in the middle of the existing view, not the entire image

Implements [markupItem](#).

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupVLine.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/markupVLine.cpp

9.42 message_types Class Reference

Public Member Functions

- **message_types** (message_severities severityIn, message_kind_sets kind_setIn=MESSAGE_KIND_STANDARD)
- QString [getSeverityName](#) ()

Function to provide string name for each message type severity.

Public Attributes

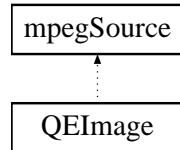
- message_severities **severity**
- message_kind_sets **kind_set**

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

- /tmp/epicsqt/trunk/framework/widgets/include/UserMessage.h
- /tmp/epicsqt/trunk/framework/widgets/src/UserMessage.cpp

9.43 mpegSource Class Reference

Inheritance diagram for mpegSource:



Public Member Functions

- void [updateImage \(FFBuffer *buf\)](#)
- void [setURL \(QString\)](#)
- void [startStream \(\)](#)
- void [stopStream \(\)](#)

Protected Member Functions

- QString [getURL \(\)](#)
- void [setURL \(QString urlIn\)](#)
- void [stopStream \(\)](#)
- void [startStream \(\)](#)

9.43.1 Member Function Documentation

9.43.1.1 void mpegSource::updateImage (FFBuffer * buf)

!???? * 3 for color only

!! Since the [QEImage](#) widget handles (or should handle) CA image data in all the formats that are expected in this mpeg stream !! perhaps this formatting here should be simply packaging the data in a QByteArray and delivering it, rather than perform any conversion.

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/mpeg.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/QEImage.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/mpeg.cpp

9.44 mpegSourceObject Class Reference

Public Slots

- void [updateImage \(FFBuffer *buf\)](#)

Signals

- void **aboutToQuit** ()

Public Member Functions

- **mpegSourceObject** ([mpegSource](#) *msIn)
- void **sentAboutToQuit** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/mpeg.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/mpeg.cpp

9.45 QEStripChartToolBar::OwnWidgets Class Reference

Public Member Functions

- **OwnWidgets** ([QEStripChartToolBar](#) *parent)

Public Attributes

- QPushButtons * **pushButtons** [NUMBER_OF_BUTTONS]
- QLabel * **yScaleStatus**
- QLabel * **timeStatus**

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartToolBar.cpp

9.46 PeriodicDialog Class Reference

Public Member Functions

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

Protected Member Functions

- void **changeEvent** (QEvent *e)

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/PeriodicDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/PeriodicDialog.cpp

9.47 PeriodicElementSetupForm Class Reference

Public Member Functions

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

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/PeriodicElementSetupForm.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/PeriodicElementSetupForm.cpp

9.48 PeriodicSetupDialog Class Reference

Public Member Functions

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

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/PeriodicSetupDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/PeriodicSetupDialog.cpp

9.49 PersistanceManager Class Reference

Public Member Functions

- QObject * **getSaveRestoreObject** ()
- void **save** (const QString fileName, const QString rootName, const QString configName)
- void **restore** (const QString fileName, const QString rootName, const QString configName)
- bool **isRestoring** ()
- **PMElement** **addNamedConfiguration** (QString name)
- **PMElement** **getNamedConfiguration** (QString name)

- QStringList **getConfigNames** (QString fileName, QString rootName)
- QStringList **getConfigNames** (QString fileName, QString rootName, bool &hasDefault)
- void **deleteConfigs** (QString fileName, QString rootName, QStringList names)

Public Attributes

- bool **restoring**

Static Public Attributes

- static QString **defaultName**

Friends

- class **PMElement**

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

- /tmp/epicsqt/trunk/framework/widgets/include/persistManager.h
- /tmp/epicsqt/trunk/framework/widgets/src/persistManager.cpp

9.50 playbackTimer Class Reference

Public Member Functions

- **playbackTimer** ([recording](#) *recorderIn)
- void **timerEvent** (QTimerEvent *event)

Public Attributes

- [recording](#) * **recorder**

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/recording.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/recording.cpp

9.51 PMContext Class Reference

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

- /tmp/epicsqt/trunk/framework/widgets/include/persistManager.h

9.52 PMElement Class Reference

Public Member Functions

- **PMElement** ([PersistanceManager](#) *ownerIn, QDomElement elementIn)
- **PMElement addElement** (QString name)
- void **addValue** (QString name, bool value)
- void **addValue** (QString name, int value)
- void **addValue** (QString name, double value)
- void **addValue** (QString name, QString value)
- void **addAttribute** (QString name, bool value)
- void **addAttribute** (QString name, int value)
- void **addAttribute** (QString name, double value)
- void **addAttribute** (QString name, QString value)
- **PMElement getElement** (QString name)
- **PMElement getElement** (QString name, int i)
- **PMElement getElement** (QString name, QString attrName, QString attrValue)
- **PMElement getElement** (QString name, QString attrName, int attrValue)
- **PMElementList getElementList** (QString name)
- bool **getValue** (QString name, bool &val)
- bool **getValue** (QString name, int &val)
- bool **getValue** (QString name, double &val)
- bool **getValue** (QString name, QString &val)
- bool **getAttribute** (QString name, bool &val)
- bool **getAttribute** (QString name, int &val)
- bool **getAttribute** (QString name, double &val)
- bool **getAttribute** (QString name, QString &val)
- bool **isNull** ()

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

- /tmp/epicsqt/trunk/framework/widgets/include/persistanceManager.h
- /tmp/epicsqt/trunk/framework/widgets/src/persistanceManager.cpp

9.53 PMElementList Class Reference

Public Member Functions

- **PMElementList** ([PersistanceManager](#) *ownerIn, QDomNodeList elementListIn)
- **PMElement getElement** (int i)
- int **count** ()

9.53.1 Member Function Documentation

9.53.1.1 PMElement PMElementList::getElement (int *i*)

!! check range of i

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

- /tmp/epicsqt/trunk/framework/widgets/include/persistanceManager.h
- /tmp/epicsqt/trunk/framework/widgets/src/persistanceManager.cpp

9.54 pointInfo Class Reference

Public Member Functions

- void **setX** (long *x*)
- void **setY** (long *y*)
- void **setPoint** (QPoint *pln*)
- void **clearX** ()
- void **clearY** ()
- bool **getStatus** ()
- QPoint **getPoint** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/QEImage.h

9.55 processManager Class Reference

Public Slots

- void **doRead** ()
- void **doReadToStandardOutput** ()
- void **doReadToStandardError** ()
- void **doFinished** (int, QProcess::ExitStatus)

Signals

- void **processCompleted** ()

Public Member Functions

- **processManager** (bool logOutput, bool useStandardIo, QTemporaryFile *tempFileIn)

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

- /tmp/epicsqt/trunk/framework/widgets/include/applicationLauncher.h
- /tmp/epicsqt/trunk/framework/widgets/src/applicationLauncher.cpp

9.56 profilePlot Class Reference

Public Types

- enum **plotDirections** { PROFILEPLOT_LR, PROFILEPLOT_RL, PROFILEPLOT_TB, PROFILEPLOT_BT }

Public Member Functions

- **profilePlot** (plotDirections plotDirectionIn)
- void **setProfile** (QVector< QPointF > *profile, double minX, double maxX, double minY, double maxY, QString title, QPoint start, QPoint end, unsigned int thicknessIn)
- void **clearProfile** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/profilePlot.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/profilePlot.cpp

9.57 PublishedProfile Class Reference

Public Attributes

- QObject * **guiLaunchConsumer**
- QStringList **pathList**
- QString **parentPath**
- QList< QString > **macroSubstitutions**
- unsigned int **messageFormId**
- QList< WidgetRef > **containedWidgets**
- **userLevelSignal** **userSignal**
- QString **userLevelPassword**
- QString **scientistLevelPassword**
- QString **engineerLevelPassword**

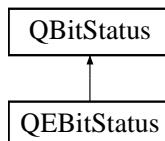
- bool **profileDefined**
- **PersistanceManager persistanceManager**
- bool **dontActivateYet**
- bool **userLevelPasswordsSet**

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

- /tmp/epicsqt/trunk/framework/widgets/include/ContainerProfile.h

9.58 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** ()

Protected Member Functions

- void **setisActive** (const bool value)
- bool **getisActive** ()

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**

- bool **isActive**

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

- /tmp/epicsqt/trunk/framework/widgets/QEBitStatus/QBitStatus.h
- /tmp/epicsqt/trunk/framework/widgets/QEBitStatus/QBitStatus.cpp

9.59 QCaAlarmInfo Class Reference

Public Member Functions

- **QCaAlarmInfo** (unsigned short statusIn, unsigned short severityIn)
- bool **operator==** (const [QCaAlarmInfo](#) &other) const
- bool **operator!=** (const [QCaAlarmInfo](#) &other) const
- QString **statusName** () const
- QString **severityName** () const
- bool **isInAlarm** () const
- bool **isMinor** () const
- bool **isMajor** () const
- bool **isValid** () const
- QString **style** () const
- QString **getColorName** () const
- QCAALARMINFO_SEVERITY **getSeverity** () const

Static Public Member Functions

- static QCAALARMINFO_SEVERITY **getInvalidSeverity** ()

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

- /tmp/epicsqt/trunk/framework/data/include/QCaAlarmInfo.h
- /tmp/epicsqt/trunk/framework/data/src/QCaAlarmInfo.cpp

9.60 QCaConnectionInfo Class Reference

Public Member Functions

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

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

- /tmp/epicsqt/trunk/framework/data/include/QCaConnectionInfo.h
- /tmp/epicsqt/trunk/framework/data/src/QCaConnectionInfo.cpp

9.61 QCaDataPoint Class Reference

Public Member Functions

- bool **isDisplayable** () const
- QString **toString** () const
- QString **toString** (const QCaDateTime &originDateTime) const

Public Attributes

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

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

- /tmp/epicsqt/trunk/framework/data/include/QCaDataPoint.h
- /tmp/epicsqt/trunk/framework/data/src/QCaDataPoint.cpp

9.62 QCaDataPointList Class Reference

Public Member Functions

- void **clear** ()
- void **removeLast** ()
- void **removeFirst** ()
- void **append** (const QCaDataPointList &other)
- void **append** (const QCaDataPoint &r)
- void **replace** (int i, const QCaDataPoint &t)
- int **count** () const
- QCaDataPoint **value** (const int j) const
- QCaDataPoint **last** () const
- void **resample** (const QCaDataPointList &source, const double interval, const QCaDateTime &endTime)
- void **compact** (const QCaDataPointList &source)
- void **toStream** (QTextStream &target, bool withIndex, bool withRelativeTime) const

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

- /tmp/epicsqt/trunk/framework/data/include/QCaDataPoint.h
- /tmp/epicsqt/trunk/framework/data/src/QCaDataPoint.cpp

9.63 QCaDateTime Class Reference

Public Member Functions

- **QCaDateTime** (QDateTime dt)
- **QCaDateTime & operator=** (const **QCaDateTime** &other)
- **QCaDateTime** (unsigned long seconds, unsigned long nanoseconds)
- **QString text** ()
- double **floating** (const QDateTime &base) const
- unsigned long **getSeconds** () const
 - Recover original EPICS time constructor parameters.*
- unsigned long **getNanoSeconds** () const

9.63.1 Member Function Documentation

9.63.1.1 double QCaDateTime::floating (const QDateTime & base) const

Duration in seconds from base time to this time. Note: this is the opposite sense to the parent QDateTime daysTo, secsTo and msecsTo functions.

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

- /tmp/epicsqt/trunk/framework/data/include/QCaDateTime.h
- /tmp/epicsqt/trunk/framework/data/src/QCaDateTime.cpp

9.64 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:

- /tmp/epicsqt/trunk/framework/data/include/QCaEventFilter.h
- /tmp/epicsqt/trunk/framework/data/src/QCaEventFilter.cpp

9.65 QCaEventItem Class Reference

Public Member Functions

- **QCaEventItem** ([QCaEventUpdate](#) *newEvent)

Public Attributes

- [QCaEventUpdate](#) * **event**

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

- /tmp/epicsqt/trunk/framework/data/include/QCaEventUpdate.h

9.66 QCaEventUpdate Class Reference

Public Member Functions

- **QCaEventUpdate** ([qcaobject::QCObject](#) *emitterObjectIn, long newReason, void *newDataPtr)

Public Attributes

- bool **acceptThisEvent**
- [qcaobject::QCObject](#) * **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:

- /tmp/epicsqt/trunk/framework/data/include/QCaEventUpdate.h
- /tmp/epicsqt/trunk/framework/data/src/QCaEventUpdate.cpp

9.67 QCInstalledFiltersListItem Class Reference

Public Member Functions

- **QCInstalledFiltersListItem** (QObject *eventObjectIn)

Public Attributes

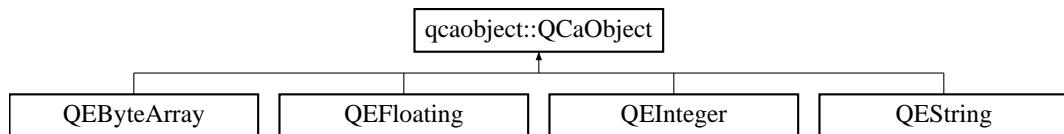
- `QObject * eventObject`
- `long referenceCount`

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

- `/tmp/epicsql/trunk/framework/data/include/QCaEventFilter.h`

9.68 qcaobject::QCaObject Class Reference

Inheritance diagram for qcaobject::QCaObject:



Public Types

- enum `priorities` { `QE_PRIORITY_LOW`, `QE_PRIORITY_NORMAL`, `QE_PRIORITY_HIGH` }

Public Slots

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

Signals

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

Public Member Functions

- `QCaObject (const QString &recordName, QObject *eventObject, const unsigned int variableIndex, unsigned char signalsToSendIn=SIG_VARIANT, priorities priorityIn=QE_PRIORITY_NORMAL)`

- **QCaObject** (const QString &recordName, QObject *eventObject, const unsigned int variableIndex, [UserMessage](#) *userMessageIn, unsigned char signalsToSendIn=SIG_VARIANT, priorities priorityIn=QE_PRIORITY_NORMAL)
- bool **subscribe** ()
- bool **singleShotRead** ()
- bool **dataTypeKnown** ()
- unsigned int **getVariableIndex** () const
- 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** ()
- void **setRequestedElementCount** (unsigned int elementCount)
- QString **getRecordName** ()
- QString **getEgu** ()
- QStringList **getEnumerations** ()
- unsigned int **getPrecision** ()
- [QCAlarmInfo](#) **getAlarmInfo** ()
- [QCaDateTime](#) **getDateTime** ()
- double **getDisplayLimitUpper** ()
- double **getDisplayLimitLower** ()
- double **getAlarmLimitUpper** ()
- double **getAlarmLimitLower** ()
- double **getWarningLimitUpper** ()
- double **getWarningLimitLower** ()
- double **getControlLimitUpper** ()
- double **getControlLimitLower** ()
- generic::generic_types **getDataType** ()
- QString **getHostName** ()
- QString **getFieldType** ()
- unsigned long **getElementCount** ()
- void **getLastData** (bool &isDefined, QVariant &value, [QCAlarmInfo](#) &alarmInfo, [QCaDateTime](#) &timeStamp)

Static Public Member Functions

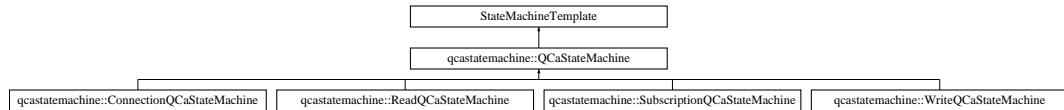
- static void **processEventStatic** ([QCEventUpdate](#) *dataUpdateEvent)

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

- /tmp/epicsqt/trunk/framework/data/include/QCaObject.h
- /tmp/epicsqt/trunk/framework/data/src/QCaObject.cpp

9.69 qcastatemachine::QCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::QCaStateMachine:



Public Member Functions

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

Public Attributes

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

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

- /tmp/epicsqt/trunk/framework/data/include/QCaStateMachine.h
- /tmp/epicsqt/trunk/framework/data/src/QCaStateMachine.cpp

9.70 QCaVariableNamePropertyManager Class Reference

Signals

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

Public Member Functions

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

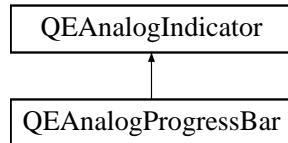
The documentation for this class was generated from the following files:

- /tmp/epicsqt/trunk/framework/data/include/QCaVariableNamePropertyManager.h
- /tmp/epicsqt/trunk/framework/data/src/QCaVariableNamePropertyManager.cpp

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

- QColor `getBackgroundColour () const`
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 () const`
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 () const`
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 () const`
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 () const`
Access function for `logScale` - refer to `logScale` property for details.

Protected Member Functions

- virtual QString `getTextImage ()`
- virtual BandList `getBandList ()`
- void `setIsActive (const bool value)`
- bool `getIsActive () const`

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`
- bool `isActive`

Alternative to isEnabled. Default is true.

9.71.1 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.

9.71.2 Member Enumeration Documentation

9.71.2.1 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)

9.71.2.2 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.

9.71.3 Property Documentation

9.71.3.1 QColor QEAnalogIndicator::backgroundColour [read, write]

Background colour

9.71.3.2 QColor QEAnalogIndicator::borderColour [read, write]

Border colour

9.71.3.3 int QEAnalogIndicator::centreAngle [read, write]

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

9.71.3.4 QColor QEAnalogIndicator::fontColour [read, write]

Font colour

9.71.3.5 QColor QEAnalogIndicator::foregroundColour [read, write]

Foreground colour

9.71.3.6 bool QEAnalogIndicator::logScale [read, write]

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

9.71.3.7 int QEAnalogIndicator::logScaleInterval [read, write]

Log scale interval.

9.71.3.8 double QEAnalogIndicator::majorInterval [read, write]

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

9.71.3.9 double QEAnalogIndicator::maximum [read, write]

Maximum indicated value.

9.71.3.10 double QEAnalogIndicator::minimum [read, write]

Minimum indicated value.

9.71.3.11 double QEAnalogIndicator::minorInterval [read, write]

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

9.71.3.12 Modes QEAnalogIndicator::mode [read, write]

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

9.71.3.13 Orientations QEAnalogIndicator::orientation [read, write]

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

9.71.3.14 bool QEAnalogIndicator::showScale [read, write]

If set, show the scale

9.71.3.15 bool QEAnalogIndicator::showText [read, write]

If set, show textual representation of value on the indicator

9.71.3.16 int QEAnalogIndicator::spanAngle [read, write]

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

9.71.3.17 double QEAnalogIndicator::value [read, write]

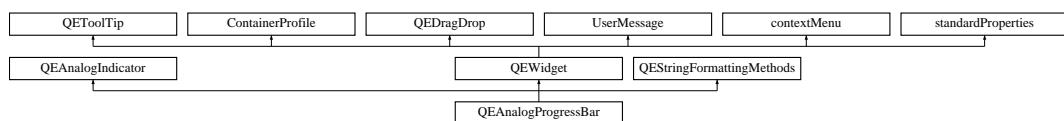
Current indicated value.

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

- /tmp/epicsqt/trunk/framework/widgets/QEAnalogIndicator/QEAnalogIndicator.h
 - /tmp/epicsqt/trunk/framework/widgets/QEAnalogIndicator/QEAnalogIndicator.cpp

9.72 QEAnalogProgressBar Class Reference

Inheritance diagram for QEAnalogProgressBar:



Public Types

- enum **UserLevels** { **User** = userLevelTypes::USERLEVEL_USER, **Scientist** = userLevelTypes::USERLEVEL_SCIENTIST, **Engineer** = userLevelTypes::USERLEVEL_ENGINEER }
 - enum **AlarmSeverityDisplayModes** { **foreground**, **background** }
 - enum **Formats** {
 Default = QStringFormatting::FORMAT_DEFAULT, **Floating** = QStringFormatting::FORMAT_FLOATING, **Integer** = QStringFormatting::FORMAT_INTEGER, **UnsignedInteger** = QStringFormatting::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 = QEString-
Formatting::ASCII, Index = QEStringFormatting::INDEX }

```

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

- **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 **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 **setAlarmSeverityDisplayStyle** (AlarmSeverityDisplayModes value)
Access function for `#AlarmSeverityDisplayStyle` property - refer to `#AlarmSeverityDisplayStyle` property for details.
- AlarmSeverityDisplayModes **getAlarmSeverityDisplayStyle** ()
Access function for `#AlarmSeverityDisplayStyle` property - refer to `#AlarmSeverityDisplayStyle` 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 **allowDrop**
- bool **visible**
- unsigned **int**
- QString **userLevelUserStyle**
- QString **userLevelScientistStyle**
- QString **userLevelEngineerStyle**
- UserLevels **userLevelVisibility**
- UserLevels **userLevelEnabled**

- bool [displayAlarmState](#)
- AlarmSeverityDisplayModes [alarmSeverityDisplayMode](#)
- bool [useDbDisplayLimits](#)
- int [precision](#)
- bool [useDbPrecision](#)
- bool [leadingZero](#)
- bool [trailingZeros](#)
- bool [addUnits](#)
- QString [localEnumeration](#)
- [Formats format](#)
- [Notations notation](#)
- [ArrayActions arrayAction](#)

9.72.1 Member Enumeration Documentation

9.72.1.1 enum QEAnalogProgressBar::ArrayActions

User friendly enumerations for arrayAction property - refer to [QEStringFormatting::arrayActions](#) for details.

Enumerator:

- Append** Refer to [QEStringFormatting::APPEND](#) for details.
Ascii Refer to [QEStringFormatting::ASCII](#) for details.
Index Refer to [QEStringFormatting::INDEX](#) for details.

9.72.1.2 enum QEAnalogProgressBar::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.

9.72.1.3 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.

9.72.1.4 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.

9.72.2 Constructor & Destructor Documentation

9.72.2.1 QEAnalogProgressBar::QEAnalogProgressBar (*QWidget * parent = 0*)

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

9.72.2.2 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.

9.72.3 Member Function Documentation

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

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

9.72.4 Property Documentation

9.72.4.1 bool QEAnalogProgressBar::addUnits [read, write]

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

9.72.4.2 AlarmSeverityDisplayModes QEAnalogProgressBar::alarmSeverityDisplayStyle [read, write]

Visualise the EPICS alarm severity

9.72.4.3 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](#).

9.72.4.4 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'.

9.72.4.5 bool QEAnalogProgressBar::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [standardProperties](#).

9.72.4.6 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).

9.72.4.7 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.

9.72.4.8 bool QEAnalogProgressBar::leadingZero [read, write]

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

9.72.4.9 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)
>= Greater 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"

9.72.4.10 Notations QEAnalogProgressBar::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.72.4.11 int QEAnalogProgressBar::precision [read, write]

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

9.72.4.12 bool QEAnalogProgressBar::trailingZeros [read, write]

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

9.72.4.13 bool QEAnalogProgressBar::useDbDisplayLimits [read, write]

Use the EPICS database display limits

9.72.4.14 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.

9.72.4.15 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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.72.4.16 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 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.

9.72.4.17 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 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.

9.72.4.18 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.

9.72.4.19 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 programmatically 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'.

9.72.4.20 QString QEAnalogProgressBar::variable [read, write]

EPICS variable name (CA PV)

9.72.4.21 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](#).

9.72.4.22 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 also used for other purposes.

9.72.4.23 bool QEAnalogProgressBar::visible [read, write]

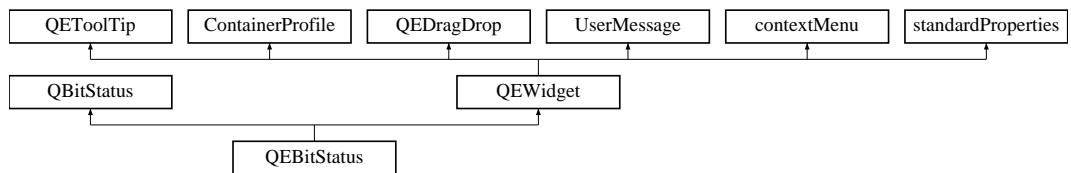
Display the widget. Default is true. Setting this property false is useful if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEAnalogProgressBar/QEAnalogProgressBar.h
 - /tmp/epicsqt/trunk/framework/widgets/QEAnalogProgressBar/QEAnalogProgressBar.cpp

9.73 QEBitStatus Class Reference

Inheritance diagram for QEBitStatus:



Public Types

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

Signals

- void **dbValueChanged** (const long &out)

Public Member Functions

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

9.73.1 Member Enumeration Documentation

9.73.1.1 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.

9.73.2 Member Function Documentation

9.73.2.1 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.

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

Virtual function that may be implemented by users of [QEWidget](#) to update variable names and macro substitutions. A default is provided that is suitable in most cases.

Reimplemented from [QEWidget](#).

9.73.3 Property Documentation

9.73.3.1 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](#).

9.73.3.2 bool QEBitStatus::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [standardProperties](#).

9.73.3.3 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.

9.73.3.4 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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.73.3.5 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 `styleManager` class. Refer to the `styleManager` class for details about how this Style Sheet string will be merged with any pre-existing Style Sheet string and any Style Sheet strings generated during the display of data.

9.73.3.6 QString QEBitStatus::userLevelScientistStyle [read, write]

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

9.73.3.7 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.

9.73.3.8 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 programmatically through `setUserLevel()`. Widgets that are always visible should be visible at 'User'. Widgets that are only used by scientists managing the facility should be visible at 'Scientist'. Widgets that are only used by engineers maintaining the facility should be visible at 'Engineer'.

9.73.3.9 QString QEBitStatus::variable [read, write]

EPICS variable name (CA PV)

9.73.3.10 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](#).

9.73.3.11 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 also used for other purposes.

9.73.3.12 bool QEBitStatus::visible [read, write]

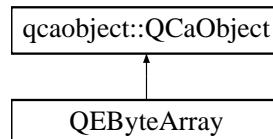
Display the widget. Default is true. Setting this property false is useful if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEBitStatus/QEBitStatus.h
- /tmp/epicsqt/trunk/framework/widgets/QEBitStatus/QEBitStatus.cpp

9.74 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

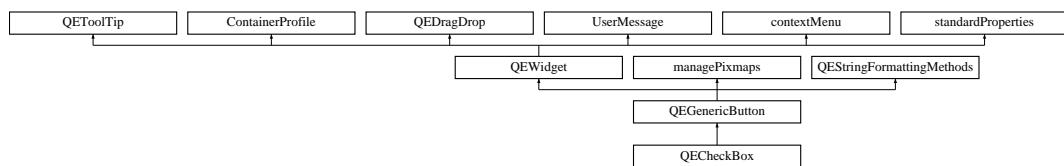
- **QEByteArray** (QString recordName, QObject *eventObject, unsigned int variableIndexIn)
- **QEByteArray** (QString recordName, QObject *eventObject, unsigned int variableIndexIn, UserMessage *userMessageIn)

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

- /tmp/epicsqt/trunk/framework/data/include/QEByteArray.h
- /tmp/epicsqt/trunk/framework/data/src/QEByteArray.cpp

9.75 QECheckBox Class Reference

Inheritance diagram for QECheckBox:



Public Types

- enum **UserLevels** { **User** = userLevelTypes::USERLEVEL_USER, **Scientist** = userLevelTypes::USERLEVEL_SCIENTIST, **Engineer** = userLevelTypes::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` = QEGenericButton::UPDATE_TEXT, `Icon` = QEGenericButton::UPDATE_ICON, `TextAndIcon` = QEGenericButton::UPDATE_TEXT_AND_ICON, `State` = QEGenericButton::UPDATE_STATE }
- User friendly enumerations for updateOption property - refer to QEGenericButton::updateOptions for details.*
- enum `ProgramStartupOptionNames` { `None` = applicationLauncher::PSO_NONE, `Terminal` = applicationLauncher::PSO_TERMINAL, `LogOutput` = applicationLauncher::PSO_LOGOUTPUT, `StdOutput` = applicationLauncher::PSO_STDOUTPUT }
 - enum `CreationOptionNames` {

`Open` = QEActionRequests::OptionOpen, `NewTab` = QEActionRequests::OptionNewTab,

`NewWindow` = QEActionRequests::OptionNewWindow, `DockTop` = QEActionRequests::OptionTopDockWindow,

`DockBottom` = QEActionRequests::OptionBottomDockWindow, `DockLeft` = QEActionRequests::OptionLeftDockWindow, `DockRight` = QEActionRequests::OptionRightDockWindow,

`DockTopTabbed` = QEActionRequests::OptionTopDockWindowTabbed,

`DockBottomTabbed` = QEActionRequests::OptionBottomDockWindowTabbed, `DockLeftTabbed` = QEActionRequests::OptionLeftDockWindowTabbed, `DockRightTabbed` = QEActionRequests::OptionRightDockWindowTabbed, `DockFloating` = QEActionRequests::OptionFloatingDockWindow }

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 `requestAction` (const QEActionRequests &request)

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` (const QEActionRequests &request)

Internal use only. Request a new GUI is created. Typically, this is caught by the QEGui application.
- void `pressed` (int value)
- void `released` (int value)
- void `clicked` (int value)
- void `programCompleted` ()

Program started by button has completed.

Public Member Functions

- `QECheckBox` (QWidget *parent=0)
- `QECheckBox` (const QString &variableName, QWidget *parent=0)

- `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 allowDrop`
- `bool visible`
- `unsigned int`
- `QString userLevelUserStyle`
- `QString userLevelScientistStyle`
- `QString userLevelEngineerStyle`
- `UserLevels userLevelVisibility`
- `UserLevels userLevelEnabled`
- `bool displayAlarmState`
- `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`
- `QString confirmText`
- `bool writeOnPress`
- `bool writeOnRelease`
- `bool writeOnClick`
- `QString pressText`
- `QString releaseText`
- `QString clickText`
- `QString clickCheckedText`
- `QString labelText`
- `QString program`
- `QStringList arguments`
- `ProgramStartupOptionNames programStartupOption`
- `QString guiFile`
- `CreationOptionNames creationOption`
- `QString prioritySubstitutions`
- `QString customisationName`

9.75.1 Member Enumeration Documentation

9.75.1.1 enum QECheckBox::ArrayActions

User friendly enumerations for arrayAction property - refer to [QEStringFormatting::arrayActions](#) for details.

Enumerator:

- Append** Refer to [QEStringFormatting::APPEND](#) for details.
Ascii Refer to [QEStringFormatting::ASCII](#) for details.
Index Refer to [QEStringFormatting::INDEX](#) for details.

9.75.1.2 enum QECheckBox::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.
- DockTop*** Open new GUI in a top dock window.
- DockBottom*** Open new GUI in a bottom dock window.
- DockLeft*** Open new GUI in a left dock window.
- DockRight*** Open new GUI in a right dock window.
- DockTopTabbed*** Open new GUI in a top dock window (tabbed with any existing dock in that area)
- DockBottomTabbed*** Open new GUI in a bottom dock window (tabbed with any existing dock in that area)
- DockLeftTabbed*** Open new GUI in a left dock window (tabbed with any existing dock in that area)
- DockRightTabbed*** Open new GUI in a right dock window (tabbed with any existing dock in that area)
- DockFloating*** Open new GUI in a floating dock window.

9.75.1.3 enum QECheckBox::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.

9.75.1.4 enum QECheckBox::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.

9.75.1.5 enum QECheckBox::ProgramStartupOptionNames

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

Enumerator:

None Just run the program.

Terminal Run the program in a terminal (in Windows a command interpreter will also be started, so the program may be a built-in command like 'dir')

LogOutput Run the program, and log the output in the QE message system.

StdOutput Run the program, and send output to standard output and standard error.

9.75.1.6 enum QECheckBox::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)

9.75.1.7 enum QECheckBox::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.

9.75.2 Constructor & Destructor Documentation

9.75.2.1 `QECheckBox::QECheckBox (QWidget * parent = 0)`

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

9.75.2.2 `QECheckBox::QECheckBox (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.

9.75.3 Member Function Documentation

9.75.3.1 `void QECheckBox::clicked (int value) [signal]`

Button has been Clicked. The value emitted is the integer interpretation of the `clickText` property (or the `clickCheckedText` property if the button was checked)

9.75.3.2 `void QECheckBox::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.

9.75.3.3 `void QECheckBox::pressed (int value) [signal]`

Button has been Pressed. The value emitted is the integer interpretation of the `pressText` property

9.75.3.4 `void QECheckBox::released (int value) [signal]`

Button has been Released The value emitted is the integer interpretation of the `releaseText` property

9.75.3.5 `void QECheckBox::requestAction (const QEActionRequests & request) [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.

9.75.4 Property Documentation

9.75.4.1 bool QECheckBox::addUnits [read, write]

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

9.75.4.2 Qt::Alignment QECheckBox::alignment [read, write]

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

9.75.4.3 bool QECheckBox::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](#).

9.75.4.4 QStringList QECheckBox::arguments [read, write]

Arguments for program specified in the 'program' property.

9.75.4.5 ArrayActions QECheckBox::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'.

9.75.4.6 QString QECheckBox::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 display 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 display 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](#).

9.75.4.7 QString QECheckBox::clickText [read, write]

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

Reimplemented from [QEGenericButton](#).

9.75.4.8 bool QECheckBox::confirmAction [read, write]

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

9.75.4.9 QString QECheckBox::confirmText [read, write]

Text used to confirm action if confirmation dialog is presented

Reimplemented from [QEGenericButton](#).

9.75.4.10 CreationOptionNames QECheckBox::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](#).

9.75.4.11 QString QECheckBox::customisationName [read, write]

Window customisation name. This name will be used to select a set of window customisations including menu items and tool bar buttons. Applications such as QEGui

can load .xml files containing named sets of window customisations. This property is used to select a set loaded from these files. The selected set of customisations will be applied to the main window containing the new GUI.

Reimplemented from [QEGenericButton](#).

9.75.4.12 bool QECheckBox::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [standardProperties](#).

9.75.4.13 Formats QECheckBox::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).

9.75.4.14 QString QECheckBox::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::openQEFfile\(\)](#) in QEWidget.cpp for details.

9.75.4.15 unsigned QECheckBox::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.

9.75.4.16 QString QECheckBox::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. IF NOT LEFT EMPTY, THIS TEXT WILL TAKE PRIORITY OVER THE PUSH BUTTON 'text' PROPERTY! 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](#).

9.75.4.17 bool QECheckBox::leadingZero [read, write]

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

9.75.4.18 QString QECheckBox::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)
>= Greater 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:"Beamlne 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"

9.75.4.19 Notations QECheckBox::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.75.4.20 QString QECheckBox::password [read, write]

Password user will need to enter before any action is taken

Reimplemented from [QEGenericButton](#).

9.75.4.21 QPixmap QECheckBox:: pixmap0 [read, write]

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

9.75.4.22 QPixmap QECheckBox:: pixmap1 [read, write]

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

9.75.4.23 QPixmap QECheckBox:: pixmap2 [read, write]

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

9.75.4.24 QPixmap QECheckBox:: pixmap3 [read, write]

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

9.75.4.25 QPixmap QECheckBox:: pixmap4 [read, write]

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

9.75.4.26 QPixmap QECheckBox:: pixmap5 [read, write]

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

9.75.4.27 QPixmap QECheckBox:: pixmap6 [read, write]

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

9.75.4.28 QPixmap QECheckBox:: pixmap7 [read, write]

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

9.75.4.29 int QECheckBox::precision [read, write]

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

9.75.4.30 QString QECheckBox::pressText [read, write]

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

Reimplemented from [QEGenericButton](#).

9.75.4.31 QString QECheckBox::prioritySubstitutions [read, write]

Overriding macro substitutions. These macro substitions take precedence over any existing macro substitutions defined by the variableSubstitutions property, any parent forms, or the application containing the button. These macro substitutions are particularly usefull when the button's function is to reload the same form but with different macro substitutions. The variableSubstitutions property cannot be used for this since, although they are added to the list of macro substitions applied to the new form, they are appended to the list and the existing macro substitutions take precedence.

Reimplemented from [QEGenericButton](#).

9.75.4.32 QString QECheckBox::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

9.75.4.33 ProgramStartupOptionNames QECheckBox::programStartupOption [read, write]

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

9.75.4.34 QString QECheckBox::releaseText [read, write]

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

Reimplemented from [QEGenericButton](#).

9.75.4.35 bool QECheckBox::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](#).

9.75.4.36 bool QECheckBox::trailingZeros [read, write]

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

9.75.4.37 UpdateOptions QECheckBox::updateOption [read, write]

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

Reimplemented from [QEGenericButton](#).

9.75.4.38 bool QECheckBox::useDbPrecision [read, write]

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

9.75.4.39 UserLevels QECheckBox::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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.75.4.40 QString QECheckBox::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.

9.75.4.41 QString QECheckBox::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.

9.75.4.42 QString QECheckBox::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.

9.75.4.43 UserLevels QECheckBox::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 programmatically 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'.

9.75.4.44 QString QECheckBox::variable [read, write]

EPICS variable name (CA PV)

9.75.4.45 bool QECheckBox::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](#).

9.75.4.46 QString QECheckBox::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 also used for other purposes.

9.75.4.47 bool QECheckBox::visible [read, write]

Display the widget. Default is true. Setting this property false is useful 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.

9.75.4.48 bool QECheckBox::writeOnClick [read, write]

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

Reimplemented from [QEGenericButton](#).

9.75.4.49 **bool QECheckBox::writeOnPress** [read, write]

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

Reimplemented from [QEGenericButton](#).

9.75.4.50 **bool QECheckBox::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:

- /tmp/epicsqt/trunk/framework/widgets/QEButton/QECheckBox.h
- /tmp/epicsqt/trunk/framework/widgets/QEButton/QECheckBox.cpp

9.76 QECheckBoxManager Class Reference

Public Member Functions

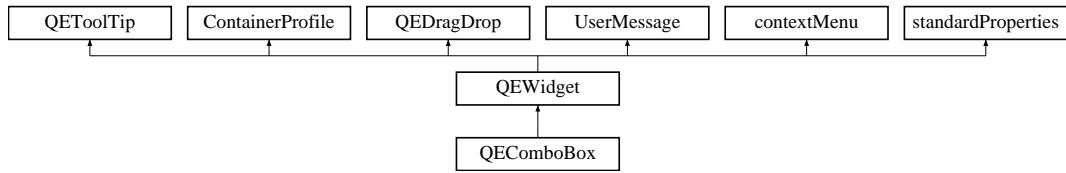
- **QECheckBoxManager** (QObject *parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget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

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

- /tmp/epicsqt/trunk/framework/widgets/QEButton/QECheckBoxManager.h
- /tmp/epicsqt/trunk/framework/widgets/QEButton/QECheckBoxManager.cpp

9.77 QEComboBox Class Reference

Inheritance diagram for QEComboBox:



Public Types

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

Signals

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

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

Public Member Functions

- `QEComboBox` (QWidget *parent=0)
- `QEComboBox` (const QString &variableName, QWidget *parent=0)
- void `setWriteOnChange` (bool writeOnChangeIn)
- bool `getWriteOnChange` ()
- void `setSubscribe` (bool subscribe)
- bool `getSubscribe` ()
- void `setUseDbEnumerations` (bool useDbEnumerations)
- bool `getUseDbEnumerations` ()
- void `setLocalEnumerations` (const QString &localEnumerations)
- QString `getLocalEnumerations` ()
- 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** ()
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)

Protected Attributes

- QEIntegerFormatting **integerFormatting**
- QELocalEnumeration **localEnumerations**
- bool **useDbEnumerations**
- bool **writeOnChange**

Properties

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

9.77.1 Member Enumeration Documentation

9.77.1.1 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.

9.77.2 Member Function Documentation

9.77.2.1 **void QEComboBox::dbValueChanged (const qulonglong & 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.

9.77.3 Member Data Documentation

9.77.3.1 **bool QEComboBox::useDbEnumerations [read, write, protected]**

Use database enumerations - defaults to true

9.77.3.2 **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).

9.77.4 Property Documentation

9.77.4.1 **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](#).

9.77.4.2 **bool QEComboBox::displayAlarmState [read, write]**

If set (default) widget will indicate the alarm state of any variable data it is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [standardProperties](#).

9.77.4.3 **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.

9.77.4.4 QString QEComboBox::localEnumeration [read, write]

Enumerations values used when useDbEnumerations is false.

9.77.4.5 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](#).

9.77.4.6 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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.77.4.7 QString QEComboBox::userLevelEngineerStyle [read, write]

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

9.77.4.8 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 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.

9.77.4.9 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.

9.77.4.10 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 programmatically 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'.

9.77.4.11 QString QEComboBox::variable [read, write]

EPICS variable name (CA PV)

9.77.4.12 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](#).

9.77.4.13 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 also used for other purposes.

9.77.4.14 bool QEComboBox::visible [read, write]

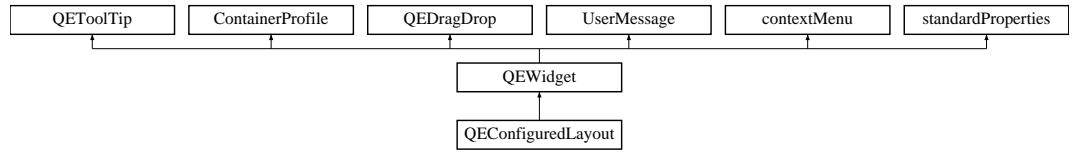
Display the widget. Default is true. Setting this property false is useful if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEComboBox/QEComboBox.h
- /tmp/epicsqt/trunk/framework/widgets/QEComboBox/QEComboBox.cpp

9.78 QEConfiguredLayout Class Reference

Inheritance diagram for QEConfiguredLayout:



Public Types

- enum **configurationTypesProperty** { **File** = FROM_FILE, **Text** = FROM_TEXT }
- enum **optionsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **userTypesProperty** { **User** = userLevelTypes::USERLEVEL_USER, **Scientist** = userLevelTypes::USERLEVEL_SCIENTIST, **Engineer** = userLevelTypes::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 **setOptionsLayout** (int pValue)
- int **getOptionsLayout** ()
- void **setCurrentUserType** (int pValue)
- int **getCurrentUserType** ()
- void **refreshFields** ()
- void **userLevelChanged** (userLevelTypes::userLevels pValue)
- void **setConfigurationTypeProperty** (configurationTypesProperty pConfigurationType)

- configurationTypesProperty **getConfigurationTypeProperty** ()
- void **setOptionsLayoutProperty** (optionsLayoutProperty pOptionsLayout)
- optionsLayoutProperty **getOptionsLayoutProperty** ()
- 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 **optionsLayout**
- int **currentUserType**
- bool **subscription**

Properties

- QString **itemDescription**
- bool **showItemList**
- configurationTypesProperty **configurationType**
- optionsLayoutProperty **optionsLayout**
- userTypesProperty **currentUserType**

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

- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayout.cpp

9.79 QEConfiguredLayoutManager Class Reference

Public Member Functions

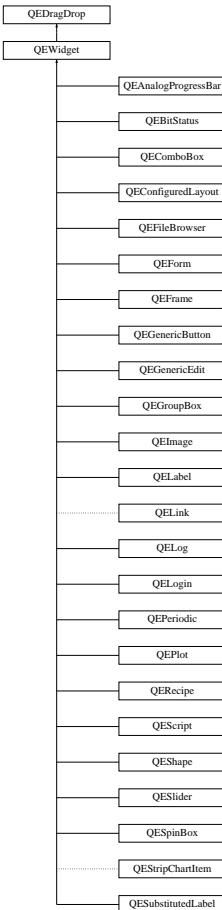
- **QEConfiguredLayoutManager** (QObject *pParent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const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

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

- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayoutManager.h
- /tmp/epicsqt/trunk/framework/widgets/QEConfiguredLayout/QEConfiguredLayoutManager.cpp

9.80 QEDragDrop Class Reference

Inheritance diagram for QEDragDrop:



Public Member Functions

- **QEDragDrop** (QWidget *ownerIn)
- bool **getAllowDrop ()**

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)

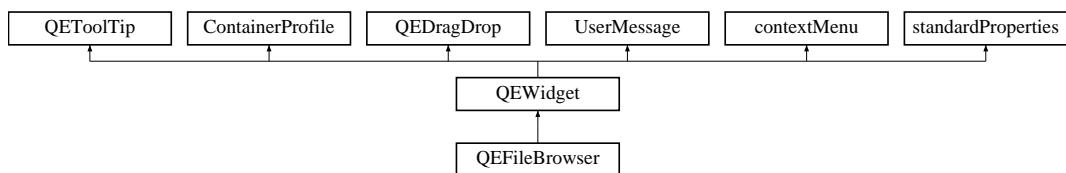
The documentation for this class was generated from the following files:

- /tmp/epicsqt/trunk/framework/widgets/include/QEDragDrop.h
- /tmp/epicsqt/trunk/framework/widgets/src/QEDragDrop.cpp

9.81 QEFileBrowser Class Reference

```
#include <QEFileBrowser.h>
```

Inheritance diagram for QEFileBrowser:



Public Types

- enum **optionsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **UserLevels** { **User** = userLevelTypes::USERLEVEL_USER, **Scientist** = userLevelTypes::USERLEVEL_SCIENTIST, **Engineer** = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void **selected** (QString pFilename)

Public Member Functions

- **QEFileBrowser** (QWidget *pParent=0)
- void **setVariableName** (QString pValue)
- QString **getVariableName** ()
- void **setVariableNameSubstitutions** (QString pValue)
- QString **getVariableNameSubstitutions** ()
- 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 **setShowTable** (bool pValue)
- bool **getShowTable** ()
- 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 **setFileDialogDirectoriesOnly** (bool pValue)
- bool **getFileDialogDirectoriesOnly** ()
- void **setOptionsLayout** (int pValue)
- int **getOptionsLayout** ()
- void **updateTable** ()
- void **setOptionsLayoutProperty** (optionsLayoutProperty pOptionsLayout)
- optionsLayoutProperty **getOptionsLayoutProperty** ()
- 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 Attributes

- QELLineEdit * **qeLineEditDirectoryPath**
- QPushButton * **qPushButtonDirectoryBrowser**
- QPushButton * **qPushButtonRefresh**
- _QTableWidgetFileBrowser * **qTableWidgetFileBrowser**
- QString **fileFilter**

*Specify which files to browse. To specify more than one filter, please separate them with a ;. Example: *py;*.ui (this will only display files with an extension .py and .ui).*
- bool **showFileExtension**

Show/hide the extension of files.
- bool **fileDialogDirectoriesOnly**

Enable/disable the browsing of directories-only when opening the dialog window.
- int **optionsLayout**

Properties

- `QString variable`
`Default directory where to browse files when QEFileBrowser is launched for the first time.`
- `QString variableSubstitutions`
- `QString directoryPath`
`Default directory where to browse files when QEFileBrowser is launched for the first time.`
- `bool showDirectoryPath`
`Show/hide directory path line edit where the user can specify the directory to browse files.`
- `bool showDirectoryBrowser`
`Show/hide button to open the dialog window to browse for directories and files.`
- `bool showRefresh`
`Show/hide button to refresh the table containing the list of files being browsed.`
- `bool showTable`
`Show/hide table containing the list of files being browsed.`
- `bool showColumnTime`
`Show/hide column containing the time of creation of files.`
- `bool showColumnSize`
`Show/hide column containing the size (in bytes) of files.`
- `bool showColumnFilename`
`Show/hide column containing the name of files.`
- `optionsLayoutProperty optionsLayout`
`Change the order of the widgets. Valid orders are: TOP, BOTTOM, LEFT and RIGHT.`
- `bool variableAsToolTip`
- `bool allowDrop`
- `bool visible`
- `unsigned int`
- `QString userLevelUserStyle`
- `QString userLevelScientistStyle`
- `QString userLevelEngineerStyle`
- `UserLevels userLevelVisibility`
- `UserLevels userLevelEnabled`
- `bool displayAlarmState`

9.81.1 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 tightly integrated with the base class `QEWidget` which provides generic support such as macro substitutions, drag/drop, and standard properties.

9.81.2 Member Enumeration Documentation

9.81.2.1 enum QEFileBrowser::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.

9.81.3 Member Function Documentation

9.81.3.1 void QEFileBrowser::selected (QString pFilename) [signal]

Signal that is generated every time the user double-clicks a certain file. This signal emits a string that contains the full path and the name of the selected file. This signal may be captured by other widgets that perform further operations (for instance, the [QEImage](#) displays the content of this file if it is a graphical one).

9.81.4 Property Documentation

9.81.4.1 bool QEFileBrowser::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](#).

9.81.4.2 bool QEFileBrowser::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [StandardProperties](#).

9.81.4.3 unsigned QEFileBrowser::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.

9.81.4.4 UserLevels QEFileBrowser::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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.81.4.5 QString QEFileBrowser::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.

9.81.4.6 QString QEFileBrowser::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.

9.81.4.7 QString QEFileBrowser::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.

9.81.4.8 UserLevels QEFileBrowser::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 programmatically 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'.

9.81.4.9 `QString QEFileBrowser::variable` [read, write]

Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
 Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
 Lorem Ipsum Lorem Ipsum Lorem Ipsum

9.81.4.10 `bool QEFileBrowser::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](#).

9.81.4.11 `QString QEFileBrowser::variableSubstitutions` [read, write]

Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
 Lorem Ipsum Lorem Ipsum Lorem Ipsum Lorem Ipsum
 Lorem Ipsum Lorem Ipsum Lorem Ipsum

9.81.4.12 `bool QEFileBrowser::visible` [read, write]

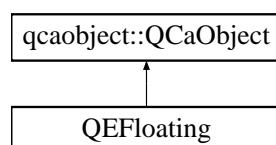
Display the widget. Default is true. Setting this property false is useful if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEFileBrowser/QEFileBrowser.h
- /tmp/epicsqt/trunk/framework/widgets/QEFileBrowser/QEFileBrowser.cpp

9.82 QEFloating Class Reference

Inheritance diagram for QEFloating:



Public Slots

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

- **QEFloatin** (QString recordName, QObject *eventObject, QEFloatinFormatting *floatingFormattingIn, unsigned int variableIndexIn)
- **QEFloatin** (QString recordName, QObject *eventObject, QEFloatinFormatting *floatingFormattingIn, unsigned int variableIndexIn, UserMessage *userMessageIn)

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

- /tmp/epicsqt/trunk/framework/data/include/QEFloatin.h
- /tmp/epicsqt/trunk/framework/data/src/QEFloatin.cpp

9.83 QEFloatinArray Class Reference

```
#include <QEFloatinArray.h>
```

Public Member Functions

- **QEFloatinArray** (int size)
- **QEFloatinArray** (int size, const double &t)
- **QEFloatinArray** (const QVector<double> &other)
- double **minimumValue** (const double &defaultValue=0.0)
- double **maximumValue** (const double &defaultValue=0.0)
- **QEFloatinArray calcDyByDx** (const QVector<double> &x)

9.83.1 Detailed Description

This class provides short hand for QVector<double> together with some basic double vector operations.

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

- /tmp/epicsqt/trunk/framework/data/include/QEFloatinArray.h
- /tmp/epicsqt/trunk/framework/data/src/QEFloatinArray.cpp

9.84 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

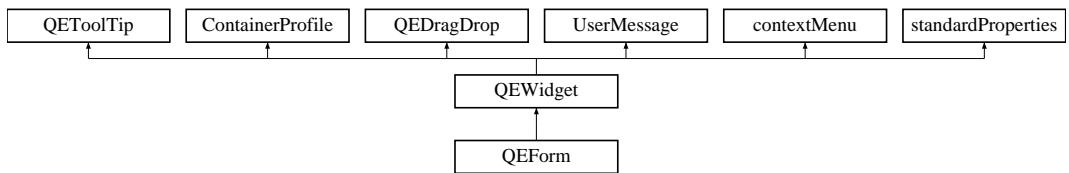
- double **formatFloating** (const QVariant &value)
- QVector< double > **formatFloatingArray** (const QVariant &value)
- QVariant **formatValue** (const double &floatingValue, generic::generic_types valueType)
- QVariant **formatValue** (const QVector< double > &floatingValue, generic::generic_types 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:

- /tmp/epicsqt/trunk/framework/data/include/QEFloatingFormatting.h
- /tmp/epicsqt/trunk/framework/data/src/QEFloatingFormatting.cpp

9.85 QEForm Class Reference

Inheritance diagram for QEForm:



Public Types

- enum **MessageFilterOptions** { **Match** = UserMessage::MESSAGE_FILTER_MATCH, **None** = UserMessage::MESSAGE_FILTER_NONE }

Public Slots

- bool **readUiFile** ()

Public Member Functions

- **QEForm** (QWidget *parent=0)
- **QEForm** (const QString &uifileNameIn, QWidget *parent=0)
- void **commonInit** (const bool alertIfUINotFoundIn)
- QString **getQEGuiTitle** ()
- QString **getFullFileName** ()
- QString **getUiFileName** ()
- void **setFileMonitoringIsEnabled** (bool fileMonitoringIsEnabled)
- bool **getFileMonitoringIsEnabled** ()
- void **setHandleGuiLaunchRequests** (bool handleGuiLaunchRequests)
- bool **getHandleGuiLaunchRequests** ()
- void **setResizeContents** (bool resizeContentsIn)
- bool **getResizeContents** ()
- QString **getContainedFrameworkVersion** ()
- QString **getUniqueIdentifier** ()
- void **setUniqueIdentifier** (QString name)
- void **setUiFileNameProperty** (QString uiFileName)
- QString **getUiFileNameProperty** ()
- void **setVariableNameSubstitutionsProperty** (QString variableNameSubstitutions)
- QString **getVariableNameSubstitutionsProperty** ()
- MessageFilterOptions **getMessageFormFilter** ()
- void **setMessageFormFilter** (MessageFilterOptions messageFormFilter)
- MessageFilterOptions **getMessageSourceFilter** ()
- void **setMessageSourceFilter** (MessageFilterOptions messageSourceFilter)

Protected Attributes

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

Properties

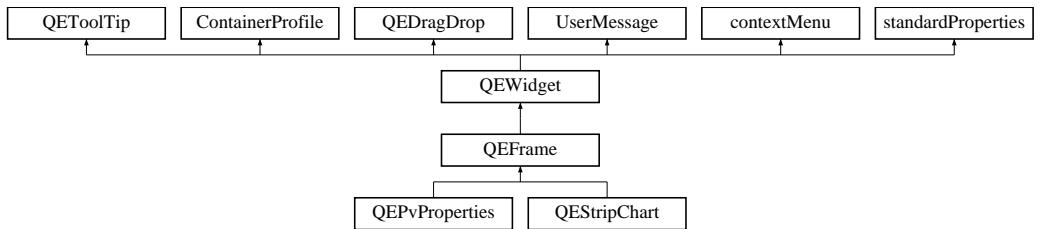
- QString **uiFile**
- QString **variableSubstitutions**
- unsigned **int**
- MessageFilterOptions **messageFormFilter**
- MessageFilterOptions **messageSourceFilter**

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

- /tmp/epicsqt/trunk/framework/widgets/QEForm/QEForm.h
- /tmp/epicsqt/trunk/framework/widgets/QEForm/QEForm.cpp

9.86 QEFrame Class Reference

Inheritance diagram for QEFrame:



Public Types

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

Public Member Functions

- `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 allowDrop`

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

9.86.1 Member Enumeration Documentation

9.86.1.1 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.

9.86.2 Property Documentation

9.86.2.1 bool QEFrame::allowDrop [read, write]

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

Reimplemented from [QEDragDrop](#).

9.86.2.2 bool QEFrame::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [standardProperties](#).

9.86.2.3 unsigned QEFrame::int [read, write]

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

9.86.2.4 UserLevels QEFrame::userLevelEnabled [read, write]

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

9.86.2.5 QString QEFrame::userLevelEngineerStyle [read, write]

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

9.86.2.6 QString QEFrame::userLevelScientistStyle [read, write]

Style Sheet string to be applied when the widget is displayed in 'Scientist' mode. Default is an empty string. The syntax is the standard Qt Style Sheet syntax. For example, 'background-color: red' This Style Sheet string will be applied by the 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.

9.86.2.7 QString QEFrame::userLevelUserStyle [read, write]

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

9.86.2.8 UserLevels QEFrame::userLevelVisibility [read, write]

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

9.86.2.9 bool QEFrame::variableAsToolTip [read, write]

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

Reimplemented from [QEToolTip](#).

9.86.2.10 bool QEFrame::visible [read, write]

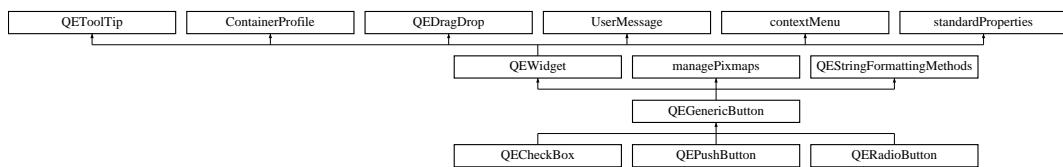
Display the widget. Default is true. Setting this property false is useful if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEFrame/QEFrame.h
- /tmp/epicsqt/trunk/framework/widgets/QEFrame/QEFrame.cpp

9.87 QEGenericButton Class Reference

Inheritance diagram for QEGenericButton:



Public Types

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

Public Member Functions

- **QEGenericButton** (QWidget *owner)
- void **setSubscribe** (bool subscribe)
- bool **getSubscribe** ()
- void **setUpdOption** (updateOptions updateOptionIn)
- updateOptions **getUpdOption** ()
- void **setTextAlignment** (Qt::Alignment alignment)
- Qt::Alignment **getTextAlignment** ()
- void **setPassword** (QString password)
- QString **getPassword** ()
- void **setConfirmAction** (bool confirmRequiredIn)

- bool **getConfirmAction** ()
- void **setConfirmText** (QString confirmTextIn)
- QString **getConfirmText** ()
- 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 **setProgramStartupOption** (applicationLauncher::programStartupOptions programStartupOptionIn)
- applicationLauncher::programStartupOptions **getProgramStartupOption** ()
- void **setGuiName** (QString guiName)
- QString **getGuiName** ()
- void **setPrioritySubstitutions** (QString prioritySubstitutionsIn)
- QString **getPrioritySubstitutions** ()
- void **setCustomisationName** (QString customisationNameIn)
- QString **getCustomisationName** ()
- void **setCreationOption** (QEActionRequests::Options creationOption)
- QEActionRequests::Options **getCreationOption** ()
- void **setLabelTextProperty** (QString labelTextIn)
- QString **getLabelTextProperty** ()

Protected Member Functions

- void **connectionChanged** (QCaConnectionInfo &connectionInfo, const unsigned int &variableIndex)
- void **setGenericButtonText** (const QString &text, QCaAlarmInfo &alarmInfo, QCaDateTime &, const unsigned int &variableIndex)
- void **userPressed** ()
- void **userReleased** ()
- void **userClicked** (bool checked)
- virtual updateOptions **getDefaultUpdateOption** ()=0
- void **startGui** (const QEActionRequests &request)
- void **setup** ()
- void **establishConnection** (unsigned int variableIndex)

Protected Attributes

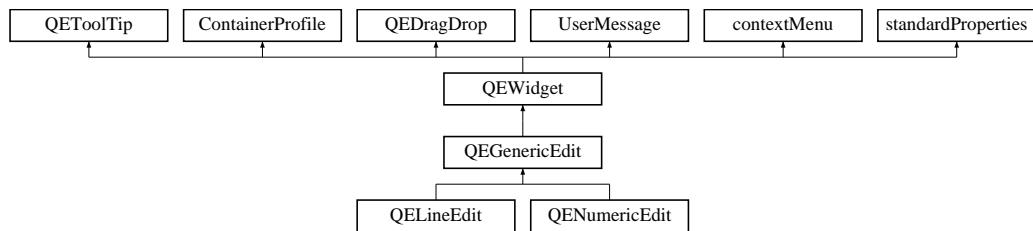
- `applicationLauncher programLauncher`

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

- /tmp/epicsqt/trunk/framework/widgets/QEBUTTON/QEGenericButton.h
- /tmp/epicsqt/trunk/framework/widgets/QEBUTTON/QEGenericButton.cpp

9.88 QEGenericEdit Class Reference

Inheritance diagram for QEGenericEdit:



Public Types

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

Signals

- void `userChange` (const QVariant &oldValue, const QVariant &newValue, const QVariant &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

- void `setVariableNameProperty` (QString variableName)

Access function for `variable` property - refer to `variable` property for details.
- QString `getVariableNameProperty` ()

Access function for `variable` property - refer to `variable` property for details.

- void [setVariableNameSubstitutionsProperty](#) (QString variableNameSubstitutions)

Access function for `variableSubstitutions` property - refer to `variableSubstitutions` property for details.
- QString [getVariableNameSubstitutionsProperty](#) ()

Access function for `variableSubstitutions` property - refer to `variableSubstitutions` 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.
- QEGenericEdit ([QWidget](#) *parent=0)
- QEGenericEdit (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](#) ()
- void [writeValue](#) ([qcaobject::QCaObject](#) *qca, QVariant newValue)
- void [writeNow](#) ()

Protected Member Functions

- void [setDataIfNoFocus](#) (const QVariant &value, [QCaAlarmInfo](#) &alarmInfo, [QCaDateTime](#) &dateTime)
- bool [getIsConnected](#) ()
- bool [testAndClearIsFirstUpdate](#) ()
- virtual void [setValue](#) (const QVariant &value)=0
- virtual QVariant [getValue](#) ()=0
- virtual bool [writeData](#) (const QVariant &value, QString &message)=0

Protected Attributes

- QVariant **lastValue**
- QVariant **lastUserValue**
- bool **messageDialogPresent**
- bool **writeFailMessageDialogPresent**
- bool **isConnected**

Properties

- QString **variable**
- QString **variableSubstitutions**
- bool **subscribe**
- bool **writeOnLoseFocus**
- bool **writeOnEnter**
- bool **writeOnFinish**
- bool **confirmWrite**
- bool **variableAsToolTip**
- bool **allowDrop**
- bool **visible**
- unsigned **int**
- QString **userLevelUserStyle**
- QString **userLevelScientistStyle**
- QString **userLevelEngineerStyle**
- UserLevels **userLevelVisibility**
- UserLevels **userLevelEnabled**
- bool **displayAlarmState**

9.88.1 Member Enumeration Documentation

9.88.1.1 enum QEGenericEdit::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.

9.88.2 Constructor & Destructor Documentation

9.88.2.1 QEGenericEdit::QEGenericEdit (QWidget * *parent* = 0)

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

```
9.88.2.2 QEGenericEdit::QEGenericEdit ( 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.

9.88.3 Member Function Documentation

```
9.88.3.1 bool QEGenericEdit::getConfirmWrite ( )
```

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

```
9.88.3.2 bool QEGenericEdit::getSubscribe ( )
```

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

```
9.88.3.3 bool QEGenericEdit::getWriteOnEnter ( )
```

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

```
9.88.3.4 bool QEGenericEdit::getWriteOnFinish ( )
```

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

```
9.88.3.5 bool QEGenericEdit::getWriteOnLoseFocus ( )
```

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

```
9.88.3.6 void QEGenericEdit::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).

```
9.88.3.7 void QEGenericEdit::setSubscribe ( bool subscribe )
```

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

9.88.3.8 void QEGenericEdit::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').

9.88.3.9 void QEGenericEdit::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).

9.88.3.10 void QEGenericEdit::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).

9.88.3.11 void QEGenericEdit::writeNow () [virtual]

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

Reimplemented from [QEWidget](#).

9.88.4 Property Documentation

9.88.4.1 bool QEGenericEdit::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](#).

9.88.4.2 bool QEGenericEdit::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).

9.88.4.3 bool QEGenericEdit::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [StandardProperties](#).

9.88.4.4 unsigned QEGenericEdit::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.

Reimplemented in [QELineEdit](#).

9.88.4.5 bool QEGenericEdit::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](#).

9.88.4.6 UserLevels QEGenericEdit::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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.88.4.7 QString QEGenericEdit::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.

9.88.4.8 QString QEGenericEdit::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.

9.88.4.9 QString QEGenericEdit::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.

9.88.4.10 UserLevels QEGenericEdit::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 programmatically 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'.

9.88.4.11 QString QEGenericEdit::variable [read, write]

EPICS variable name (CA PV)

9.88.4.12 bool QEGenericEdit::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](#).

9.88.4.13 QString QEGenericEdit::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 also used for other purposes.

9.88.4.14 bool QEGenericEdit::visible [read, write]

Display the widget. Default is true. Setting this property false is useful 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.

9.88.4.15 bool QEGenericEdit::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').

9.88.4.16 bool QEGenericEdit::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).

9.88.4.17 bool QEGenericEdit::writeOnLoseFocus [read, write]

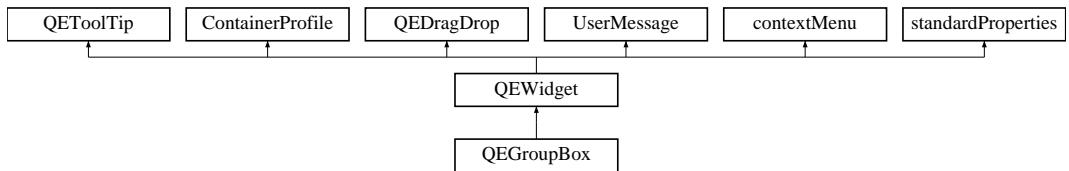
Sets if this widget automatically writes any changes when it loses focus. Default is 'false' (does not write any changes when it loses focus).

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

- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QEGenericEdit.h
- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QEGenericEdit.cpp

9.89 QEGroupBox Class Reference

Inheritance diagram for QEGroupBox:



Public Types

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

Public Member Functions

- `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`)
- `QEGroupBox` (`const QString &title, QWidget *parent=0`)
- `QSize sizeHint () const`

Protected Member Functions

- virtual void `setSubstitutionsProperty` (`QString macroSubstitutionsIn`)
- `QString getSubstitutionsProperty ()`

Properties

- bool `variableAsToolTip`
- bool `allowDrop`
- bool `visible`
- unsigned `int`
- `QString userLevelUserStyle`
- `QString userLevelScientistStyle`
- `QString userLevelEngineerStyle`
- `UserLevels userLevelVisibility`
- `UserLevels userLevelEnabled`
- bool `displayAlarmState`
- `QString substitutedTitle`
- `QString textSubstitutions`

9.89.1 Member Enumeration Documentation

9.89.1.1 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.

9.89.2 Property Documentation

9.89.2.1 bool QEGroupBox::allowDrop [read, write]

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

Reimplemented from [QEDragDrop](#).

9.89.2.2 bool QEGroupBox::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [StandardProperties](#).

9.89.2.3 unsigned QEGroupBox::int [read, write]

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

9.89.2.4 QString QEGroupBox::substitutedTitle [read, write]

Group box title text to be substituted. This text will be copied to the group box title text after applying any macro substitutions from the [textSubstitutions](#) property

9.89.2.5 QString QEGroupBox::textSubstitutions [read, write]

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

9.89.2.6 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 programmatically through [setUserLevel\(\)](#). Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.89.2.7 QString QEGroupBox::userLevelEngineerStyle [read, write]

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

9.89.2.8 QString QEGroupBox::userLevelScientistStyle [read, write]

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

9.89.2.9 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.

9.89.2.10 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 programmatically 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'.

9.89.2.11 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](#).

9.89.2.12 bool QEGroupBox::visible [read, write]

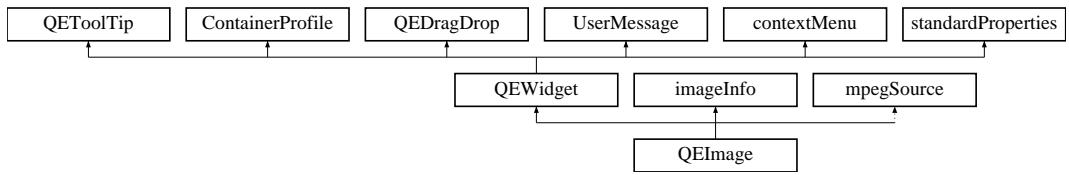
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEGroupBox/QEGroupBox.h
- /tmp/epicsqt/trunk/framework/widgets/QEGroupBox/QEGroupBox.cpp

9.90 QEImage Class Reference

Inheritance diagram for QEImage:



Public Types

- enum `selectOptions` {
 `SO_NONE`, `SO_PANNING`, `SO_VSLICE`, `SO_HSLICE`,
 `SO_AREA1`, `SO_AREA2`, `SO_AREA3`, `SO_AREA4`,
 `SO_PROFILE`, `SO_TARGET`, `SO_BEAM` }
- enum `resizeOptions` { `RESIZE_OPTION_ZOOM`, `RESIZE_OPTION_FIT` }
- enum `rotationOptions` { `ROTATION_0`, `ROTATION_90_RIGHT`, `ROTATION_90_LEFT`, `ROTATION_180` }
- enum `UserLevels` { `User` = userLevelTypes::USERLEVEL_USER, `Scientist` = userLevelTypes::USERLEVEL_SCIENTIST, `Engineer` = userLevelTypes::USERLEVEL_ENGINEER }
- enum `FormatOptions` {
 `Mono` = imageDataFormats::MONO, `Bayer` = imageDataFormats::BAYER, `rgb1` =
 imageDataFormats::RGB1, `rgb2` = imageDataFormats::RGB2,
 `rgb3` = imageDataFormats::RGB3, `yuv444` = imageDataFormats::YUV444, `yuv422` =
 imageDataFormats::YUV422, `yuv421` = imageDataFormats::YUV421 }
- enum `ResizeOptions` { `Zoom` = QEImage::RESIZE_OPTION_ZOOM, `Fit` = QEImage::RESIZE_OPTION_FIT }
- enum `RotationOptions` { `NoRotation` = QEImage::ROTATION_0, `Rotate90Right` =
 QEImage::ROTATION_90_RIGHT, `Rotate90Left` = QEImage::ROTATION_90_LEFT,
 `Rotate180` = QEImage::ROTATION_180 }
- enum `ProgramStartupOptionNames` { `None` = applicationLauncher::PSO_NONE,
 `Terminal` = applicationLauncher::PSO_TERMINAL, `LogOutput` = applicationLauncher::PSO_LOGOUTPUT,
 `StdOutput` = applicationLauncher::PSO_STDOUPUT }

Public Slots

- void **setImageFile** (QString name)
• void **setSelectPanMode** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **setSelectVSliceMode** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **setSelectHSliceMode** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **setSelectArea1Mode** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **setSelectArea2Mode** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **setSelectArea3Mode** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **setSelectArea4Mode** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **setSelectProfileMode** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **setSelectTargetMode** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **setSelectBeamMode** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **pauseClicked** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **saveClicked** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **targetClicked** ()
Framework use only. Slot to allow external setting of selection menu options.
- void **imageDisplayPropsDestroyed** (QObject *)
Framework use only. Slot to catch deletion of components (such as profile plots) that have been passed to the application for presentation.
- void **vSliceDisplayDestroyed** (QObject *)
Framework use only. Slot to catch deletion of components (such as profile plots) that have been passed to the application for presentation.
- void **hSliceDisplayDestroyed** (QObject *)
Framework use only. Slot to catch deletion of components (such as profile plots) that have been passed to the application for presentation.
- void **profileDisplayDestroyed** (QObject *)
Framework use only. Slot to catch deletion of components (such as profile plots) that have been passed to the application for presentation.
- void **recorderDestroyed** (QObject *)
Framework use only. Slot to catch deletion of components (such as profile plots) that have been passed to the application for presentation.

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 **componentHostRequest** (const QEActionRequests &request)

Public Member Functions

- **QEImage** (QWidget *parent=0)
- **QEImage** (const QString &variableName, QWidget *parent=0)
- **~QEImage** ()
Destructor.
- **selectOptions getSelectionOption** ()
- void **setBitDepth** (unsigned int bitDepthIn)
Access function for #bitDepth property - refer to #bitDepth property for details.
- unsigned int **getBitDepth** ()
Access function for #bitDepth property - refer to #bitDepth property for details.
- void **setFormatOption** (imageDataFormats::formatOptions formatOption)
Access function for formatOption property - refer to formatOption property for details.
- imageDataFormats::formatOptions **getFormatOption** ()
Access function for formatOption property - refer to formatOption property for details.
- void **setResizeOption** (resizeOptions resizeOptionIn)
Access function for #resizeOption property - refer to #resizeOption property for details.
- **resizeOptions getResizeOption** ()
Access function for #resizeOption property - refer to #resizeOption property for details.
- void **setZoom** (int zoomIn)
Access function for zoom property - refer to zoom property for details.
- int **getZoom** ()
Access function for zoom property - refer to zoom property for details.
- void **setRotation** (rotationOptions rotationIn)
Access function for #rotation property - refer to #rotation property for details.
- **rotationOptions getRotation** ()
Access function for #rotation property - refer to #rotation property for details.
- void **setHorizontalFlip** (bool flipHozIn)
Access function for horizontalFlip property - refer to horizontalFlip property for details.
- bool **getHorizontalFlip** ()
Access function for horizontalFlip property - refer to horizontalFlip property for details.
- void **setVerticalFlip** (bool flipVertIn)
Access function for verticalFlip property - refer to verticalFlip property for details.
- bool **getVerticalFlip** ()
Access function for verticalFlip property - refer to verticalFlip property for details.
- void **setInitialHozScrollPos** (int initialHosScrollPosIn)

- Access function for `initialHozScrollPos` property - refer to `initialHozScrollPos` property for details.
 - int `getInitialHozScrollPos ()`
Access function for `initialHozScrollPos` property - refer to `initialHozScrollPos` property for details.
 - void `setInitialVertScrollPos (int initialVertScrollPosIn)`
Access function for `initialVertScrollPos` property - refer to `initialVertScrollPos` property for details.
 - int `getInitialVertScrollPos ()`
Access function for `initialVertScrollPos` property - refer to `initialVertScrollPos` property for details.
 - void `setDisplayButtonBar (bool displayButtonBarIn)`
Access function for `displayButtonBar` property - refer to `displayButtonBar` property for details.
 - bool `getDisplayButtonBar ()`
Access function for `displayButtonBar` property - refer to `displayButtonBar` property for details.
 - void `setShowTime (bool pValue)`
Access function for `showTime` property - refer to `showTime` property for details.
 - bool `getShowTime ()`
Access function for `showTime` property - refer to `showTime` property for details.
 - void `setUseFalseColour (bool pValue)`
Access function for `useFalseColour` property - refer to `useFalseColour` property for details.
 - bool `getUseFalseColour ()`
Access function for `useFalseColour` property - refer to `useFalseColour` property for details.
 - void `setVertSliceMarkupColor (QColor pValue)`
Access function for `vertSliceColor` property - refer to `vertSliceColor` property for details.
 - QColor `getVertSliceMarkupColor ()`
Access function for `vertSliceColor` property - refer to `vertSliceColor` property for details.
 - void `setHozSliceMarkupColor (QColor pValue)`
Access function for `hozSliceColor` property - refer to `hozSliceColor` property for details.
 - QColor `getHozSliceMarkupColor ()`
Access function for `hozSliceColor` property - refer to `hozSliceColor` property for details.
 - void `setProfileMarkupColor (QColor pValue)`
Access function for `profileColor` property - refer to `profileColor` property for details.
 - QColor `getProfileMarkupColor ()`
Access function for `profileColor` property - refer to `profileColor` property for details.
 - void `setAreaMarkupColor (QColor pValue)`
Access function for `areaColor` property - refer to `areaColor` property for details.
 - QColor `getAreaMarkupColor ()`
Access function for `areaColor` property - refer to `areaColor` property for details.
 - void `setTargetMarkupColor (QColor pValue)`
Access function for `targetColor` property - refer to `targetColor` property for details.

- QColor **getTargetMarkupColor ()**
Access function for *targetColor* property - refer to *targetColor* property for details.
- void **setBeamMarkupColor (QColor pValue)**
Access function for *beamColor* property - refer to *beamColor* property for details.
- QColor **getBeamMarkupColor ()**
Access function for *beamColor* property - refer to *beamColor* property for details.
- void **setTimeMarkupColor (QColor pValue)**
Access function for *timeColor* property - refer to *timeColor* property for details.
- QColor **getTimeMarkupColor ()**
Access function for *timeColor* property - refer to *timeColor* property for details.
- void **setEllipseMarkupColor (QColor markupColor)**
Access function for *ellipseColor* property - refer to *ellipseColor* property for details.
- QColor **getEllipseMarkupColor ()**
Access function for *ellipseColor* property - refer to *ellipseColor* property for details.
- void **setDisplayCursorPixelInfo (bool displayCursorPixelInfo)**
Access function for *displayCursorPixelInfo* property - refer to *displayCursorPixelInfo* property for details.
- bool **getDisplayCursorPixelInfo ()**
Access function for *displayCursorPixelInfo* property - refer to *displayCursorPixelInfo* property for details.
- void **setContrastReversal (bool contrastReversalIn)**
Access function for *contrastReversal* property - refer to *contrastReversal* property for details.
- bool **getContrastReversal ()**
Access function for *contrastReversal* property - refer to *contrastReversal* property for details.
- void **setLog (bool log)**
Access function for *logBrightness* property - refer to *logBrightness* property for details.
- bool **getLog ()**
Access function for *logBrightness* property - refer to *logBrightness* property for details.
- void **setEnableVertSliceSelection (bool enableVSliceSelection)**
Access function for *enableVertSliceSelection* property - refer to *enableVertSliceSelection* property for details.
- bool **getEnableVertSliceSelection ()**
Access function for *enableVertSliceSelection* property - refer to *enableVertSliceSelection* property for details.
- void **setEnableHozSliceSelection (bool enableHSliceSelection)**
Access function for *enableHozSliceSelection* property - refer to *enableHozSliceSelection* property for details.
- bool **getEnableHozSliceSelection ()**
Access function for *enableHozSliceSelection* property - refer to *enableHozSliceSelection* property for details.
- void **setEnableArea1Selection (bool enableAreaSelectionIn)**
Access function for *enableArea1Selection* property - refer to *enableArea1Selection* property for details.

- `bool getEnableArea1Selection ()`
Access function for `enableArea1Selection` property - refer to `enableArea1Selection` property for details.
- `void setEnableArea2Selection (bool enableAreaSelectionIn)`
Access function for `enableArea2Selection` property - refer to `enableArea2Selection` property for details.
- `bool getEnableArea2Selection ()`
Access function for `enableArea2Selection` property - refer to `enableArea2Selection` property for details.
- `void setEnableArea3Selection (bool enableAreaSelectionIn)`
Access function for `enableArea3Selection` property - refer to `enableArea3Selection` property for details.
- `bool getEnableArea3Selection ()`
Access function for `enableArea3Selection` property - refer to `enableArea3Selection` property for details.
- `void setEnableArea4Selection (bool enableAreaSelectionIn)`
Access function for `enableArea4Selection` property - refer to `enableArea4Selection` property for details.
- `bool getEnableArea4Selection ()`
Access function for `enableArea4Selection` property - refer to `enableArea4Selection` property for details.
- `void setEnableProfileSelection (bool enableProfileSelectionIn)`
Access function for `enableProfileSelection` property - refer to `enableProfileSelection` property for details.
- `bool getEnableProfileSelection ()`
Access function for `enableProfileSelection` property - refer to `enableProfileSelection` property for details.
- `void setEnableTargetSelection (bool enableTargetSelectionIn)`
Access function for `enableTargetSelection` property - refer to `enableTargetSelection` property for details.
- `bool getEnableTargetSelection ()`
Access function for `enableTargetSelection` property - refer to `enableTargetSelection` property for details.
- `void setEnableBeamSelection (bool enableBeamSelectionIn)`
Access function for `enableBeamSelection` property - refer to `enableBeamSelection` property for details.
- `bool getEnableBeamSelection ()`
Access function for `enableBeamSelection` property - refer to `enableBeamSelection` property for details.
- `void setEnableImageDisplayProperties (bool enableImageDisplayPropertiesIn)`
Access function for `enableImageDisplayProperties` property - refer to `enableImageDisplayProperties` property for details.
- `bool getEnableImageDisplayProperties ()`
Access function for `enableImageDisplayProperties` property - refer to `enableImageDisplayProperties` property for details.
- `void setEnableRecording (bool enableRecordingIn)`

- `bool getEnableRecording ()`
Access function for `enableRecording` property - refer to `enableRecording` property for details.
- `void setAutoBrightnessContrast (bool autoBrightnessContrastIn)`
Access function for `autoBrightnessContrast` property - refer to `autoBrightnessContrast` property for details.
- `bool getAutoBrightnessContrast ()`
Access function for `autoBrightnessContrast` property - refer to `autoBrightnessContrast` property for details.
- `void setExternalControls (bool externalControlsIn)`
Access function for `externalControls` property - refer to `externalControls` property for details.
- `bool getExternalControls ()`
Access function for `externalControls` property - refer to `externalControls` property for details.
- `void setFullContextMenu (bool fullContextMenuIn)`
Access function for `#fullContextMenu` property - refer to `#fullContextMenu` property for details.
- `bool getFullContextMenu ()`
Access function for `#fullContextMenu` property - refer to `#fullContextMenu` property for details.
- `void setEnableProfilePresentation (bool enableProfilePresentationIn)`
Access function for `#enableProfilePresentation` property - refer to `#enableProfilePresentation` property for details.
- `bool getEnableProfilePresentation ()`
Access function for `#enableProfilePresentation` property - refer to `#enableProfilePresentation` property for details.
- `void setEnableHozSlicePresentation (bool enableHozSlicePresentationIn)`
Access function for `#enableHozSlicePresentation` property - refer to `#enableHozSlicePresentation` property for details.
- `bool getEnableHozSlicePresentation ()`
Access function for `#enableHozSlicePresentation` property - refer to `#enableHozSlicePresentation` property for details.
- `void setEnableVertSlicePresentation (bool enableVertSlicePresentationIn)`
Access function for `#enableVertSlicePresentation` property - refer to `#enableVertSlicePresentation` property for details.
- `bool getEnableVertSlicePresentation ()`
Access function for `#enableVertSlicePresentation` property - refer to `#enableVertSlicePresentation` property for details.
- `void setDisplayVertSliceSelection (bool displayVSliceSelection)`
Access function for `displayVertSliceSelection` property - refer to `displayVertSliceSelection` property for details.
- `bool getDisplayVertSliceSelection ()`
Access function for `displayVertSliceSelection` property - refer to `displayVertSliceSelection` property for details.

- void `setDisplayHozSliceSelection` (bool displayHSliceSelection)
Access function for `displayHozSliceSelection` property - refer to `displayHozSliceSelection` property for details.
- bool `getDisplayHozSliceSelection` ()
Access function for `displayHozSliceSelection` property - refer to `displayHozSliceSelection` property for details.
- void `setDisplayArea1Selection` (bool displayAreaSelection)
Access function for `displayArea1Selection` property - refer to `displayArea1Selection` property for details.
- bool `getDisplayArea1Selection` ()
Access function for `displayArea1Selection` property - refer to `displayArea1Selection` property for details.
- void `setDisplayArea2Selection` (bool displayAreaSelection)
Access function for `displayArea2Selection` property - refer to `displayArea2Selection` property for details.
- bool `getDisplayArea2Selection` ()
Access function for `displayArea2Selection` property - refer to `displayArea2Selection` property for details.
- void `setDisplayArea3Selection` (bool displayAreaSelection)
Access function for `displayArea3Selection` property - refer to `displayArea3Selection` property for details.
- bool `getDisplayArea3Selection` ()
Access function for `displayArea3Selection` property - refer to `displayArea3Selection` property for details.
- void `setDisplayArea4Selection` (bool displayAreaSelection)
Access function for `displayArea4Selection` property - refer to `displayArea4Selection` property for details.
- bool `getDisplayArea4Selection` ()
Access function for `displayArea4Selection` property - refer to `displayArea4Selection` property for details.
- void `setDisplayProfileSelection` (bool displayProfileSelection)
Access function for `displayProfileSelection` property - refer to `displayProfileSelection` property for details.
- bool `getDisplayProfileSelection` ()
Access function for `displayProfileSelection` property - refer to `displayProfileSelection` property for details.
- void `setDisplayTargetSelection` (bool displayTargetSelection)
Access function for `displayTargetSelection` property - refer to `displayTargetSelection` property for details.
- bool `getDisplayTargetSelection` ()
Access function for `displayTargetSelection` property - refer to `displayTargetSelection` property for details.
- void `setDisplayBeamSelection` (bool displayBeamSelection)
Access function for `displayBeamSelection` property - refer to `displayBeamSelection` property for details.
- bool `getDisplayBeamSelection` ()

- Access function for `displayBeamSelection` property - refer to `displayBeamSelection` property for details.
 - void `setDisplayEllipse` (bool displayEllipse)
Access function for `displayEllipse` property - refer to `displayEllipse` property for details.
 - bool `getDisplayEllipse` ()
Access function for `displayEllipse` property - refer to `displayEllipse` property for details.
- void `setDisplayMarkups` (bool displayMarkupsIn)
Access function for `#displayMarkups` property - refer to `#displayMarkups` property for details.
- bool `getDisplayMarkups` ()
Access function for `#displayMarkups` property - refer to `#displayMarkups` property for details.
- void `setProgram1` (QString program)
Access function for `program1` property - refer to `program1` property for details.
- QString `getProgram1` ()
Access function for `program1` property - refer to `program1` property for details.
- void `setProgram2` (QString program)
Access function for `program2` property - refer to `program2` property for details.
- QString `getProgram2` ()
Access function for `program2` property - refer to `program2` property for details.
- void `setArguments1` (QStringList arguments)
Access function for `arguments1` property - refer to `arguments1` property for details.
- QStringList `getArguments1` ()
Access function for `arguments1` property - refer to `arguments1` property for details.
- void `setArguments2` (QStringList arguments)
Access function for `arguments2` property - refer to `arguments2` property for details.
- QStringList `getArguments2` ()
Access function for `arguments2` property - refer to `arguments2` property for details.
- void `setProgramStartupOption1` (applicationLauncher::programStartupOptions programStartupOption)
Access function for `programStartupOption1` property - refer to `programStartupOption1` property for details.
- applicationLauncher::programStartupOptions `getProgramStartupOption1` ()
Access function for `programStartupOption1` property - refer to `programStartupOption1` property for details.
- void `setProgramStartupOption2` (applicationLauncher::programStartupOptions programStartupOption)
Access function for `programStartupOption2` property - refer to `programStartupOption2` property for details.
- applicationLauncher::programStartupOptions `getProgramStartupOption2` ()
Access function for `programStartupOption2` property - refer to `programStartupOption2` property for details.
- QString `getHozSliceLegend` ()
Access function for `hozSliceLegend` property - refer to `hozSliceLegend` property for details.

- void [setHozSliceLegend](#) (QString legend)
Access function for `hozSliceLegend` property - refer to `hozSliceLegend` property for details.
- QString [getVertSliceLegend](#) ()
Access function for `vertSliceLegend` property - refer to `vertSliceLegend` property for details.
- void [setVertSliceLegend](#) (QString legend)
Access function for `vertSliceLegend` property - refer to `vertSliceLegend` property for details.
- QString [getprofileLegend](#) ()
Access function for `profileLegend` property - refer to `profileLegend` property for details.
- void [setProfileLegend](#) (QString legend)
Access function for `profileLegend` property - refer to `profileLegend` property for details.
- QString [getAreaSelection1Legend](#) ()
Access function for `areaSelection1Legend` property - refer to `areaSelection1Legend` property for details.
- void [setAreaSelection1Legend](#) (QString legend)
Access function for `areaSelection1Legend` property - refer to `areaSelection1Legend` property for details.
- QString [getAreaSelection2Legend](#) ()
Access function for `areaSelection2Legend` property - refer to `areaSelection2Legend` property for details.
- void [setAreaSelection2Legend](#) (QString legend)
Access function for `areaSelection2Legend` property - refer to `areaSelection2Legend` property for details.
- QString [getAreaSelection3Legend](#) ()
Access function for `areaSelection3Legend` property - refer to `areaSelection3Legend` property for details.
- void [setAreaSelection3Legend](#) (QString legend)
Access function for `areaSelection3Legend` property - refer to `areaSelection3Legend` property for details.
- QString [getAreaSelection4Legend](#) ()
Access function for `areaSelection4Legend` property - refer to `areaSelection4Legend` property for details.
- void [setAreaSelection4Legend](#) (QString legend)
Access function for `areaSelection4Legend` property - refer to `areaSelection4Legend` property for details.
- QString [getTargetLegend](#) ()
Access function for `targetLegend` property - refer to `targetLegend` property for details.
- void [setTargetLegend](#) (QString legend)
Access function for `targetLegend` property - refer to `targetLegend` property for details.
- QString [getBeamLegend](#) ()
Access function for `beamLegend` property - refer to `beamLegend` property for details.
- void [setBeamLegend](#) (QString legend)
Access function for `beamLegend` property - refer to `beamLegend` property for details.

- `QString getEllipseLegend ()`
Access function for `ellipseLegend` property - refer to `ellipseLegend` property for details.
- `void setEllipseLegend (QString legend)`
Access function for `ellipseLegend` property - refer to `ellipseLegend` property for details.
- `bool getFullScreen ()`
Access function for `#fullScreen` property - refer to `#fullScreen` property for details.
- `void setFullScreen (bool fullScreenIn)`
Access function for `#fullScreen` property - refer to `#fullScreen` property for details.
- `void setSubstitutedUrl (QString urlIn)`
Access function for `URL` property - refer to `URL` property for details.
- `QString getSubstitutedUrl ()`
Access function for `URL` property - refer to `URL` 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)`
Access function for `formatOption` property - refer to `formatOption` property for details.
- `FormatOptions getFormatOptionProperty ()`
Access function for `formatOption` property - refer to `formatOption` property for details.
- `void setBitDepthProperty (unsigned int bitDepth)`
Access function for `#bitDepth` property - refer to `#bitDepth` property for details.
- `unsigned int getBitDepthProperty ()`
Access function for `#bitDepth` property - refer to `#bitDepth` property for details.
- `void setResizeOptionProperty (ResizeOptions resizeOption)`
Access function for `#resizeOption` property - refer to `#resizeOption` property for details.
- `ResizeOptions getResizeOptionProperty ()`
Access function for `#resizeOption` property - refer to `#resizeOption` property for details.
- `void setRotationProperty (RotationOptions rotation)`
Access function for `#rotation` property - refer to `#rotation` property for details.
- `RotationOptions getRotationProperty ()`
Access function for `#rotation` property - refer to `#rotation` property for details.
- `void setProgramStartupOptionProperty1 (ProgramStartupOptionNames1 programStartupOption)`
Access function for `#ProgramStartupOptionNames1` property - refer to `#ProgramStartupOptionNames1` property for details.

- **ProgramStartupOptionNames** `getProgramStartupOptionProperty1 ()`
Access function for #ProgramStartupOptionNames1 property - refer to #ProgramStartupOptionNames1 property for details.
- **void** `setProgramStartupOptionProperty2 (ProgramStartupOptionNames programStartupOption)`
Access function for #ProgramStartupOptionNames2 property - refer to #ProgramStartupOptionNames2 property for details.
- **ProgramStartupOptionNames** `getProgramStartupOptionProperty2 ()`
Access function for #ProgramStartupOptionNames2 property - refer to #ProgramStartupOptionNames2 property for details.

Protected Types

- enum **variableIndexes** {

IMAGE_VARIABLE, FORMAT_VARIABLE, BIT_DEPTH_VARIABLE, WIDTH_VARIABLE,

HEIGHT_VARIABLE, NUM_DIMENSIONS_VARIABLE, DIMENSION_0_VARIABLE, DIMENSION_1_VARIABLE,

DIMENSION_2_VARIABLE, ROI1_X_VARIABLE, ROI1_Y_VARIABLE, ROI1_W_VARIABLE,

ROI1_H_VARIABLE, ROI2_X_VARIABLE, ROI2_Y_VARIABLE, ROI2_W_VARIABLE,

ROI2_H_VARIABLE, ROI3_X_VARIABLE, ROI3_Y_VARIABLE, ROI3_W_VARIABLE,

ROI3_H_VARIABLE, ROI4_X_VARIABLE, ROI4_Y_VARIABLE, ROI4_W_VARIABLE,

ROI4_H_VARIABLE, TARGET_X_VARIABLE, TARGET_Y_VARIABLE, BEAM_X_VARIABLE,

BEAM_Y_VARIABLE, TARGET_TRIGGER_VARIABLE, CLIPPING_ONOFF_VARIABLE, CLIPPING_LOW_VARIABLE,

CLIPPING_HIGH_VARIABLE, PROFILE_H_VARIABLE, PROFILE_H_THICKNESS_VARIABLE, PROFILE_V_VARIABLE,

PROFILE_V_THICKNESS_VARIABLE, LINE_PROFILE_X1_VARIABLE, LINE_PROFILE_Y1_VARIABLE, LINE_PROFILE_X2_VARIABLE,

LINE_PROFILE_Y2_VARIABLE, LINE_PROFILE_THICKNESS_VARIABLE, PROFILE_H_ARRAY, PROFILE_V_ARRAY,

PROFILE_LINE_ARRAY, ELLIPSE_X1_VARIABLE, ELLIPSE_Y1_VARIABLE, ELLIPSE_X2_VARIABLE,

ELLIPSE_Y2_VARIABLE, QEIMAGE_NUM_VARIABLES }

Protected Member Functions

- **void** `establishConnection (unsigned int variableIndex)`
- **QImage** `copyImage ()`
- **void** `redisplayAllMarkups ()`
- **void** `resizeFullScreen ()`

- 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

- QEStringFormatting **stringFormatting**
- QEIntegerFormatting **integerFormatting**
- QEFloatingFormatting **floatingFormatting**
- resizeOptions **resizeOption**
- int **zoom**
Zoom percentage. Used when #resizeOption is Zoom.
- rotationOptions **rotation**
- bool **flipVert**
- bool **flipHoz**
- int **initialHozScrollPos**
- int **initialVertScrollPos**
- bool **displayButtonBar**

Properties

- QString **imageVariable**
- QString **formatVariable**
- QString **bitDepthVariable**
- QString **widthVariable**
- QString **heightVariable**
- QString **dimensionsVariable**
- QString **dimension1Variable**
- QString **dimension2Variable**
- QString **dimension3Variable**
- QString **regionOfInterest1XVariable**
- QString **regionOfInterest1YVariable**
- QString **regionOfInterest1WVariable**
- QString **regionOfInterest1HVariable**
- QString **regionOfInterest2XVariable**
- QString **regionOfInterest2YVariable**
- QString **regionOfInterest2WVariable**
- QString **regionOfInterest2HVariable**
- QString **regionOfInterest3XVariable**
- QString **regionOfInterest3YVariable**

- `QString regionOfInterest3WVariable`
- `QString regionOfInterest3HVariable`
- `QString regionOfInterest4XVariable`
- `QString regionOfInterest4YVariable`
- `QString regionOfInterest4WVariable`
- `QString regionOfInterest4HVariable`
- `QString targetXVariable`
- `QString targetYVariable`
- `QString beamXVariable`
- `QString beamYVariable`
- `QString targetTriggerVariable`
- `QString clippingOnOffVariable`
- `QString clippingLowVariable`
- `QString clippingHighVariable`
- `QString profileHozVariable`
- `QString profileHozThicknessVariable`
- `QString profileVertVariable`
- `QString profileVertThicknessVariable`
- `QString lineProfileX1Variable`
- `QString lineProfileY1Variable`
- `QString lineProfileX2Variable`
- `QString lineProfileY2Variable`
- `QString lineProfileThicknessVariable`
- `QString profileHozArrayVariable`
- `QString profileVertArrayVariable`
- `QString lineProfileArrayVariable`
- `QString ellipseX1Variable`
- `QString ellipseY1Variable`
- `QString ellipseX2Variable`
- `QString ellipseY2Variable`
- `QString variableSubstitutions`
- `bool variableAsToolTip`
- `bool allowDrop`
- `bool visible`
- `unsigned int`
- `QString userLevelUserStyle`
- `QString userLevelScientistStyle`
- `QString userLevelEngineerStyle`
- `UserLevels userLevelVisibility`
- `UserLevels userLevelEnabled`
- `bool displayAlarmState`
- `FormatOptions formatOption`
- `bool enableVertSliceSelection`
- `bool enableHozSliceSelection`
- `bool enableProfileSelection`
- `bool enableArea1Selection`

- bool `enableArea2Selection`
- bool `enableArea3Selection`
- bool `enableArea4Selection`
- bool `enableTargetSelection`
- bool `enableBeamSelection`
- QString `hozSliceLegend`
Name of horizontal slice profile markup.
- QString `vertSliceLegend`
Name of vertical slice profile markup.
- QString `profileLegend`
Name of arbitrary profile markup.
- QString `areaSelection1Legend`
Name of area selection 1 markup.
- QString `areaSelection2Legend`
Name of area selection 2 markup.
- QString `areaSelection3Legend`
Name of area selection 3 markup.
- QString `areaSelection4Legend`
Name of area selection 4 markup.
- QString `targetLegend`
Name of target markup.
- QString `beamLegend`
Name of beam markup.
- QString `ellipseLegend`
Name of ellipse markup.
- bool `displayVertSliceSelection`
- bool `displayHozSliceSelection`
- bool `displayProfileSelection`
- bool `displayArea1Selection`
- bool `displayArea2Selection`
- bool `displayArea3Selection`
- bool `displayArea4Selection`
- bool `displayTargetSelection`
- bool `displayBeamSelection`
- bool `displayEllipse`
- bool `displayCursorPixelInfo`
- bool `contrastReversal`
- bool `logBrightness`
- bool `showTime`
- bool `useFalseColour`
- QColor `vertSliceColor`
- QColor `hozSliceColor`
- QColor `profileColor`
- QColor `areaColor`
- QColor `beamColor`

- QColor `targetColor`
- QColor `timeColor`
- QColor `ellipseColor`
- `ResizeOptions resizeOption`
- `RotationOptions rotation`
- bool `verticalFlip`
- bool `horizontalFlip`
- int `initialHosScrollPos`
- bool `enableImageDisplayProperties`

If true, the local Image Display Properties controls are displayed.
- bool `enableRecording`

If true, the recording controls are displayed.
- bool `autoBrightnessContrast`
- bool `externalControls`
- bool `briefInfoArea`
- QString `program1`
- QStringList `arguments1`
- `ProgramStartupOptionNames programStartupOption1`
- QString `program2`
- QStringList `arguments2`
- `ProgramStartupOptionNames programStartupOption2`
- QString `URL`

9.90.1 Member Enumeration Documentation

9.90.1.1 enum QEImage::FormatOptions

User friendly enumerations for `formatOption` property - refer to `formatOption` property and `#formatOptions` enumeration for details.

Enumerator:

- `Mono`** Grey scale.
- `Bayer`** Colour (Bayer)
- `rgb1`** Colour (24 bit RGB)
- `rgb2`** Colour (??? bit RGB)
- `rgb3`** Colour (??? bit RGB)
- `yuv444`** Colour (???)
- `yuv422`** Colour (???)

9.90.1.2 enum QEImage::ProgramStartupOptionNames

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

Enumerator:

None Just run the program.

Terminal Run the program in a terminal (in Windows a command interpreter will also be started, so the program may be a built-in command like 'dir')

LogOutput Run the program, and log the output in the QE message system.

StdOutput Run the program, and send output to standard output and standard error.

9.90.1.3 enum QEImage::resizeOptions

Image resize options

Enumerator:

RESIZE_OPTION_ZOOM Zoom to selected percentage.

RESIZE_OPTION_FIT Zoom to fit the current window size.

9.90.1.4 enum QEImage::ResizeOptions

User friendly enumerations for #resizeOption property

Enumerator:

Zoom Zoom to selected percentage.

Fit Zoom to fit the current window size.

9.90.1.5 enum QEImage::RotationOptions

User friendly enumerations for #rotation property

Enumerator:

NoRotation No image rotation.

Rotate90Right Rotate image 90 degrees clockwise.

Rotate90Left Rotate image 90 degrees anticlockwise.

Rotate180 Rotate image 180 degrees.

9.90.1.6 enum QEImage::rotationOptions

Image rotation options

Enumerator:

- ROTATION_0*** No image rotation.
- ROTATION_90_RIGHT*** Rotate image 90 degrees clockwise.
- ROTATION_90_LEFT*** Rotate image 90 degrees anticlockwise.
- ROTATION_180*** Rotate image 180 degrees.

9.90.1.7 enum QEImage::selectOptions

Internal use only. Selection options. What will happen when the user interacts with the image area

Enumerator:

- SO_NONE*** Do nothing.
- SO_PANNING*** User is panning.
- SO_VSLICE*** Select the vertical slice point.
- SO_HSLICE*** Select the horizontal slice point.
- SO_AREA4*** User is selecting an area (for region of interest)
- SO_PROFILE*** Select an arbitrary line across the image (to determine a profile)
- SO_TARGET*** Mark the target point.
- SO_BEAM*** Mark the current beam location.

9.90.1.8 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.

9.90.2 Constructor & Destructor Documentation

9.90.2.1 QEImage::QEImage (QWidget * parent = 0)

Create without a variable. Use `setVariableName'n'Property()` - where 'n' is a number from 0 to 40 - and `setSubstitutionsProperty()` to define variables and, optionally, macro

substitutions later. Note, each variable property is named by function (such as `imageVariable` and `widthVariable`) but given a numeric get and set property access function such as `setVariableName22Property()`. Refer to the property definitions to determine what 'set' and 'get' function is used for each variable, or use Qt library functions to set or get the variable names by name.

9.90.2.2 `QEImage::QEImage (const QString & variableName, QWidget * parent = 0)`

Create with a variable. A connection is automatically established. The variable is set up as the first variable. This is consistent with other widgets, but will not result in an updating image as the width and height variables are required as a minimum.

9.90.3 Member Function Documentation

9.90.3.1 `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.

9.90.4 Member Data Documentation

9.90.4.1 `bool QEImage::displayButtonBar [read, write, protected]`

If true, a button bar will be displayed above the image. If not displayed, all buttons in the button bar are still available in the right click menu.

9.90.4.2 `int QEImage::initialVertScrollPos [read, write, protected]`

Sets the initial position of the vertical scroll bar, if present. Used to set up an initial view when zoomed in.

9.90.5 Property Documentation

9.90.5.1 `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](#).

9.90.5.2 `QColor QEImage::areaColor [read, write]`

Used to select the color of the area selection markups.

9.90.5.3 QStringList QEImage::arguments1 [read, write]

Arguments for program specified in the 'program1' property.

9.90.5.4 QStringList QEImage::arguments2 [read, write]

Arguments for program specified in the 'program2' property.

9.90.5.5 bool QEImage::autoBrightnessContrast [read, write]

If true, auto set local brightness and contrast when any area is selected. The brightness and contrast is set to use the full range of pixels in the selected area.

9.90.5.6 QColor QEImage::beamColor [read, write]

Used to select the color of the beam marker.

9.90.5.7 QString QEImage::beamXVariable [read, write]

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

9.90.5.8 QString QEImage::beamYVariable [read, write]

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

9.90.5.9 QString QEImage::bitDepthVariable [read, write]

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

9.90.5.10 bool QEImage::briefInfoArea [read, write]

If true, the information area will be brief (one row)

9.90.5.11 QString QEImage::clippingHighVariable [read, write]

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

9.90.5.12 QString QEImage::clippingLowVariable [read, write]

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

9.90.5.13 QString QEImage::clippingOnOffVariable [read, write]

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

9.90.5.14 bool QEImage::contrastReversal [read, write]

If true, the image will undergo contrast reversal.

9.90.5.15 QString QEImage::dimension1Variable [read, write]

EPICS variable name (CA PV). This variable is used to read the first area detector dimension of the image. If there are 2 dimensions, this will be the image width. If there are 3 dimensions, this will be the number of elements per pixel.

9.90.5.16 QString QEImage::dimension2Variable [read, write]

EPICS variable name (CA PV). This variable is used to read the second area detector dimension of the image. If there are 2 dimensions, this will be the image height. If there are 3 dimensions, this will be the image width.

9.90.5.17 QString QEImage::dimension3Variable [read, write]

EPICS variable name (CA PV). This variable is used to read the third area detector dimension of the image. If there are 3 dimensions, this will be the image height.

9.90.5.18 QString QEImage::dimensionsVariable [read, write]

EPICS variable name (CA PV). This variable is used to read the number of area detector dimensions of the image. If used, this will be 2 (one element per pixel arranged by width and height) or 3 (multiple elements per pixel arranged by pixel, width and height)

9.90.5.19 bool QEImage::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [StandardProperties](#).

9.90.5.20 bool QEImage::displayArea1Selection [read, write]

If true, selected area 1 will be displayed on the image. Note, this property is ignored unless the [enableArea1Selection](#) property is true.

9.90.5.21 bool QEImage::displayArea2Selection [read, write]

If true, selected area 2 will be displayed on the image. Note, this property is ignored unless the [enableArea2Selection](#) property is true.

9.90.5.22 bool QEImage::displayArea3Selection [read, write]

If true, selected area 3 will be displayed on the image. Note, this property is ignored unless the [enableArea3Selection](#) property is true.

9.90.5.23 bool QEImage::displayArea4Selection [read, write]

If true, selected area 4 will be displayed on the image. Note, this property is ignored unless the [enableArea4Selection](#) property is true.

9.90.5.24 bool QEImage::displayBeamSelection [read, write]

If true, beam selection will be displayed on the image. Note, this property is ignored unless the [enableBeamSelection](#) property is true.

9.90.5.25 bool QEImage::displayCursorPixelInfo [read, write]

If true, an area will be presented under the image with textual information about the pixel under the cursor, and for other selections such as selected areas.

9.90.5.26 bool QEImage::displayEllipse [read, write]

If true, the ellipse markup will be displayed on the image.

9.90.5.27 bool QEImage::displayHozSliceSelection [read, write]

If true, the selected horizontal slice will be displayed on the image. Note, this property is ignored unless the [enableHozSliceSelection](#) property is true.

9.90.5.28 bool QEImage::displayProfileSelection [read, write]

If true, the selected arbitrary line will be displayed on the image. Note, this property is ignored unless the [enableProfileSelection](#) property is true.

9.90.5.29 bool QEImage::displayTargetSelection [read, write]

If true, target selection will be displayed on the image. Note, this property is ignored unless the [enableTargetSelection](#) property is true.

9.90.5.30 bool QEImage::displayVertSliceSelection [read, write]

If true, the selected vertical slice will be displayed on the image. Note, this property is ignored unless the [enableVertSliceSelection](#) property is true.

9.90.5.31 QColor QEImage::ellipseColor [read, write]

Used to select the color of the ellipse marker.

9.90.5.32 QString QEImage::ellipseX1Variable [read, write]

EPICS variable name (CA PV). This variable is used to read an ellipse start X.

9.90.5.33 QString QEImage::ellipseX2Variable [read, write]

EPICS variable name (CA PV). This variable is used to read an ellipse end X.

9.90.5.34 QString QEImage::ellipseY1Variable [read, write]

EPICS variable name (CA PV). This variable is used to read an ellipse start Y.

9.90.5.35 QString QEImage::ellipseY2Variable [read, write]

EPICS variable name (CA PV). This variable is used to read an ellipse end Y.

9.90.5.36 bool QEImage::enableArea1Selection [read, write]

If true, the user will be able to select area 1. These are used for selection of Region of Interests, and for zooming to area 1

9.90.5.37 bool QEImage::enableArea2Selection [read, write]

If true, the user will be able to select area 2. These are used for selection of Region of Interests, and for zooming to area 2

9.90.5.38 bool QEImage::enableArea3Selection [read, write]

If true, the user will be able to select area 3. These are used for selection of Region of Interests, and for zooming to area 3

9.90.5.39 bool QEImage::enableArea4Selection [read, write]

If true, the user will be able to select area 4. These are used for selection of Region of Interests, and for zooming to area 4

9.90.5.40 bool QEImage::enableBeamSelection [read, write]

If true, the user will be able to select points on the image to mark a beam position. This can be used for automatic beam positioning.

9.90.5.41 bool QEImage::enableHozSliceSelection [read, write]

If true, the option to select a horizontal slice through the image will be available to the user. This will be used to generate a horizontal pixel profile, and write the position of the slice to the optional variable specified by the [profileHozVariable](#) property. The profile will only be presented to the user if #enableHozSlicePresentation property is true.

9.90.5.42 bool QEImage::enableProfileSelection [read, write]

If true, the option to select an arbitrary line through any part of the image will be available to the user. This will be used to generate a pixel profile.

9.90.5.43 bool QEImage::enableTargetSelection [read, write]

If true, the user will be able to select points on the image to mark a target position. This can be used for automatic beam positioning.

9.90.5.44 bool QEImage::enableVertSliceSelection [read, write]

If true, the option to select a vertical slice through the image will be available to the user. This will be used to generate a horizontal pixel profile, and write the position of the slice to the optional variable specified by the [profileVertVariable](#) property. The profile will only be presented to the user if #enableVertSlicePresentation property is true.

9.90.5.45 bool QEImage::externalControls [read, write]

If true, image controls and views such as brightness controls and profile plots are hosted by the application as dock windows, toolbars, etc. Refer to the [ContainerProfile](#) class

and the `#windowCustomisation` class to see how this class asks an application to act as a host.

9.90.5.46 `FormatOptions QEImage::formatOption` [read, write]

Video format. EPICS data type size will typically be adequate for the number of bits required (one byte for 8 bits, 2 bytes for 12 and 16 bits), but can be larger (4 bytes for 24 bits.)

9.90.5.47 `QString QEImage::formatVariable` [read, write]

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

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

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

9.90.5.49 `bool QEImage::horizontalFlip` [read, write]

If true, flip image horizontally.

9.90.5.50 `QColor QEImage::hozSliceColor` [read, write]

Used to select the color of the horizontal slice markup.

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

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

9.90.5.52 `int QEImage::initialHosScrollPos` [read, write]

Sets the initial position of the horizontal scroll bar, if present. Used to set up an initial view when zoomed in.

9.90.5.53 `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.

Bit depth. Note, EPICS data type size will typically be adequate for the number of bits required (one byte for up to 8 bits, 2 bytes for up to 16 bits, etc), but can be larger (for example, 4 bytes for 24 bits) and may be larger than necessary (4 bytes for 8 bits).

9.90.5.54 `QString QEImage::lineProfileArrayVariable` [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector arbitrary line profile array.

9.90.5.55 `QString QEImage::lineProfileThicknessVariable` [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector arbitrary line profile end Y.

9.90.5.56 `QString QEImage::lineProfileX1Variable` [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector arbitrary line profile start X.

9.90.5.57 `QString QEImage::lineProfileX2Variable` [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector arbitrary line profile end X.

9.90.5.58 `QString QEImage::lineProfileY1Variable` [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector arbitrary line profile start Y.

9.90.5.59 `QString QEImage::lineProfileY2Variable` [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector arbitrary line profile end Y.

9.90.5.60 `bool QEImage::logBrightness` [read, write]

If true, the image will be displayed using a logarithmic brightness scale.

9.90.5.61 `QColor QEImage::profileColor` [read, write]

Used to select the color of the arbitrary profile line markup.

9.90.5.62 QString QEImage::profileHozArrayVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector horizontal profile array.

9.90.5.63 QString QEImage::profileHozThicknessVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector horizontal profile thickness.

9.90.5.64 QString QEImage::profileHozVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector horizontal profile.

9.90.5.65 QString QEImage::profileVertArrayVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector vertical profile array.

9.90.5.66 QString QEImage::profileVertThicknessVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector vertical profile.

9.90.5.67 QString QEImage::profileVertVariable [read, write]

EPICS variable name (CA PV). This variable is used to write the areadetector vertical profile.

9.90.5.68 QString QEImage::program1 [read, write]

Program to run when a request is made to pass on the current image to the first external application. No attempt to run a program is made if this property is empty. Example: paint.exe

9.90.5.69 QString QEImage::program2 [read, write]

Program to run when a request is made to pass on the current image to the second external application. No attempt to run a program is made if this property is empty. Example: paint.exe

9.90.5.70 ProgramStartupOptionNames QEImage::programStartupOption1 [read, write]

Startup options for the program specified in the 'program1' property. Just run the command, run the command within a terminal, or display the output in QE message system.

9.90.5.71 ProgramStartupOptionNames QEImage::programStartupOption2 [read, write]

Startup options for the program specified in the 'program2' property. Just run the command, run the command within a terminal, or display the output in QE message system.

9.90.5.72 QString QEImage::regionOfInterest1HVariable [read, write]

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

9.90.5.73 QString QEImage::regionOfInterest1WVariable [read, write]

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

9.90.5.74 QString QEImage::regionOfInterest1XVariable [read, write]

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

9.90.5.75 QString QEImage::regionOfInterest1YVariable [read, write]

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

9.90.5.76 QString QEImage::regionOfInterest2HVariable [read, write]

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

9.90.5.77 QString QEImage::regionOfInterest2WVariable [read, write]

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

9.90.5.78 QString QEImage::regionOfInterest2XVariable [read, write]

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

9.90.5.79 QString QEImage::regionOfInterest2YVariable [read, write]

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

9.90.5.80 QString QEImage::regionOfInterest3HVariable [read, write]

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

9.90.5.81 QString QEImage::regionOfInterest3WVariable [read, write]

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

9.90.5.82 QString QEImage::regionOfInterest3XVariable [read, write]

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

9.90.5.83 QString QEImage::regionOfInterest3YVariable [read, write]

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

9.90.5.84 QString QEImage::regionOfInterest4HVariable [read, write]

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

9.90.5.85 QString QEImage::regionOfInterest4WVariable [read, write]

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

9.90.5.86 QString QEImage::regionOfInterest4XVariable [read, write]

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

9.90.5.87 QString QEImage::regionOfInterest4YVariable [read, write]

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

9.90.5.88 ResizeOptions QEImage::resizeOption [read, write]

Resize option. Zoom to zoom to the percentage given by the [zoom](#) property, or fit to the window size.

9.90.5.89 RotationOptions QEImage::rotation [read, write]

Image rotation option.

9.90.5.90 bool QEImage::showTime [read, write]

If true, the image timestamp will be written in the top left of the image.

9.90.5.91 QColor QEImage::targetColor [read, write]

Used to select the color of the target marker.

9.90.5.92 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.

9.90.5.93 QString QEImage::targetXVariable [read, write]

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

9.90.5.94 QString QEImage::targetYVariable [read, write]

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

9.90.5.95 QColor QEImage::timeColor [read, write]

Used to select the color of the timestamp.

9.90.5.96 QString QEImage::URL [read, write]

MPEG stream URL. If this is specified, this will be used as the source of the image in preference to variables (variables defining the image data, width, and height will be ignored)

9.90.5.97 bool QEImage::useFalseColour [read, write]

If true, the apply false colour to the image.

9.90.5.98 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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.90.5.99 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 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.

9.90.5.100 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 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.

9.90.5.101 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.

9.90.5.102 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 programmatically 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'.

9.90.5.103 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](#).

9.90.5.104 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.

9.90.5.105 bool QEImage::verticalFlip [read, write]

If true, flip image vertically.

9.90.5.106 QColor QEImage::vertSliceColor [read, write]

Used to select the color of the vertical slice markup.

9.90.5.107 bool QEImage::visible [read, write]

Display the widget. Default is true. Setting this property false is useful 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.

9.90.5.108 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:

- /tmp/epicsqt/trunk/framework/widgets/QEImage/QEImage.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/QEImage.cpp

9.91 QEImageMarkupThickness Class Reference

Public Member Functions

- **QEImageMarkupThickness** (QWidget *parent=0)
- void **setThickness** (unsigned int thicknessIn)
- unsigned int **getThickness** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/QEImageMarkupThickness.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/QEImageMarkupThickness.cpp

9.92 QEImageOptionsDialog Class Reference

Signals

- void **optionChange** (imageContextMenu::imageContextMenuOptions option, bool checked)

Public Member Functions

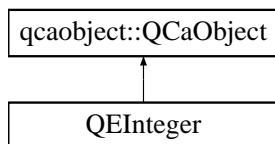
- **QEImageOptionsDialog** (QWidget *parent=0)
- void **initialise** ()
- void **optionSet** (imageContextMenu::imageContextMenuOptions option, bool checked)
- bool **optionGet** (imageContextMenu::imageContextMenuOptions option)

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/QEImageOptionsDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/QEImageOptionsDialog.cpp

9.93 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:

- /tmp/epicsqt/trunk/framework/data/include/QEInteger.h
- /tmp/epicsqt/trunk/framework/data/src/QEInteger.cpp

9.94 QEIntegerArray Class Reference

```
#include <QEIntegerArray.h>
```

Public Member Functions

- **QEIntegerArray** (int size)
- **QEIntegerArray** (int size, const long &t)
- **QEIntegerArray** (const QVector< long > &other)
- long **minimumValue** (const long &defaultValue=0)
- long **maximumValue** (const long &defaultValue=0)

9.94.1 Detailed Description

This class provides short hand for QVector<long> together with some basic long vector operations.

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

- /tmp/epicsqt/trunk/framework/data/include/QEIntegerArray.h
- /tmp/epicsqt/trunk/framework/data/src/QEIntegerArray.cpp

9.95 QEIntegerFormatting Class Reference

```
#include <QEIntegerFormatting.h>
```

Public Member Functions

- [QEIntegerFormatting \(\)](#)
Constructor.
- long [formatInteger \(const QVariant &value\)](#)
- QVector< long > [formatIntegerArray \(const QVariant &value\)](#)
- QVariant [formatValue \(const long &integerValue, generic::generic_types valueType\)](#)
- void [setRadix \(unsigned int radix\)](#)
Set the radix used for all conversions. Default is 10.
- unsigned int [getPrecision \(\)](#)
Get the precision used for all conversions.
- unsigned int [getRadix \(\)](#)
Get the radix used for all conversions.

9.95.1 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 its 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.

9.95.2 Member Function Documentation

9.95.2.1 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.

9.95.2.2 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 QCObject, 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.

9.95.2.3 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 QCObject.

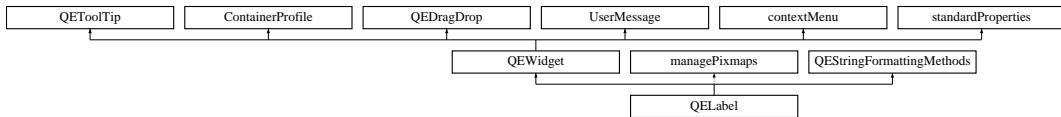
The documentation for this class was generated from the following files:

- /tmp/epicsqt/trunk/framework/data/include/QEIntegerFormatting.h
- /tmp/epicsqt/trunk/framework/data/src/QEIntegerFormatting.cpp

9.96 QELabel Class Reference

```
#include <QELabel.h>
```

Inheritance diagram for QELabel:



Public Types

- enum `updateOptions` { `UPDATE_TEXT`, `UPDATE_PIXMAP` }
- enum `UserLevels` { `User` = `userLevelTypes::USERLEVEL_USER`, `Scientist` = `userLevelTypes::USERLEVEL_SCIENTIST`, `Engineer` = `userLevelTypes::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.

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)
- `UserLevels getUserLevelVisibilityProperty` ()
Access function for `userLevelVisibility` property - refer to `userLevelVisibility` property for details.
- `void setUserLevelVisibilityProperty` (UserLevels level)
Access function for `userLevelVisibility` property - refer to `userLevelVisibility` property for details.
- `UserLevels getUserLevelEnabledProperty` ()
Access function for `userLevelEnabled` property - refer to `userLevelEnabled` property for details.
- `void setUserLevelEnabledProperty` (UserLevels level)
Access function for `userLevelEnabled` property - refer to `userLevelEnabled` property for details.
- `void setFormatProperty` (Formats format)
Access function for `format` property - refer to `format` property for details.
- `Formats getFormatProperty` ()
Access function for `format` property - refer to `format` property for details.
- `void setNotationProperty` (Notations notation)
Access function for `notation` property - refer to `notation` property for details.
- `Notations getNotationProperty` ()
Access function for `notation` property - refer to `notation` property for details.
- `void setArrayActionProperty` (ArrayActions arrayAction)
Access function for `arrayAction` property - refer to `arrayAction` property for details.
- `ArrayActions getArrayActionProperty` ()
Access function for `arrayAction` property - refer to `arrayAction` property for details.
- `void setUpdateOptionProperty` (UpdateOptions updateOption)
Access function for #updateOption property - refer to #updateOption property for details.

- **UpdateOptions getUpdateOptionProperty ()**
Access function for #updateOption property - refer to #updateOption property for details.
- void **setPixmap0Property (QPixmap pixmap)**
'Set' access function for pixmap0 properties. Refer to pixmap0 property for details
- void **setPixmap1Property (QPixmap pixmap)**
'Set' access function for pixmap1 properties. Refer to pixmap1 property for details
- void **setPixmap2Property (QPixmap pixmap)**
'Set' access function for pixmap2 properties. Refer to pixmap2 property for details
- void **setPixmap3Property (QPixmap pixmap)**
'Set' access function for pixmap3 properties. Refer to pixmap3 property for details
- void **setPixmap4Property (QPixmap pixmap)**
'Set' access function for pixmap4 properties. Refer to pixmap4 property for details
- void **setPixmap5Property (QPixmap pixmap)**
'Set' access function for pixmap5 properties. Refer to pixmap5 property for details
- void **setPixmap6Property (QPixmap pixmap)**
'Set' access function for pixmap6 properties. Refer to pixmap6 property for details
- void **setPixmap7Property (QPixmap pixmap)**
'Set' access function for pixmap7 properties. Refer to pixmap7 property for details
- QPixmap **getPixmap0Property ()**
'Get' access function for pixmap0 properties. Refer to pixmap0 property for details
- QPixmap **getPixmap1Property ()**
'Get' access function for pixmap1 properties. Refer to pixmap1 property for details
- QPixmap **getPixmap2Property ()**
'Get' access function for pixmap2 properties. Refer to pixmap2 property for details
- QPixmap **getPixmap3Property ()**
'Get' access function for pixmap3 properties. Refer to pixmap3 property for details
- QPixmap **getPixmap4Property ()**
'Get' access function for pixmap4 properties. Refer to pixmap4 property for details
- QPixmap **getPixmap5Property ()**
'Get' access function for pixmap5 properties. Refer to pixmap5 property for details
- QPixmap **getPixmap6Property ()**
'Get' access function for pixmap6 properties. Refer to pixmap6 property for details
- QPixmap **getPixmap7Property ()**
'Get' access function for pixmap7 properties. Refer to pixmap7 property for details

Properties

- QString **variable**
- QString **variableSubstitutions**
- bool **variableAsToolTip**
- bool **allowDrop**
- bool **visible**

- `unsigned int`
- `QString userLevelUserStyle`
- `QString userLevelScientistStyle`
- `QString userLevelEngineerStyle`
- `UserLevels userLevelVisibility`
- `UserLevels userLevelEnabled`
- `bool displayAlarmState`
- `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`

9.96.1 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 tightly integrated with the base class [QEWidget](#) which provides generic support such as macro substitutions, drag/drop, and standard properties.

9.96.2 Member Enumeration Documentation

9.96.2.1 enum QELabel::ArrayActions

User friendly enumerations for arrayAction property - refer to [QEStringFormatting::arrayActions](#) for details.

Enumerator:

- Append** Refer to [QEStringFormatting::APPEND](#) for details.
Ascii Refer to [QEStringFormatting::ASCII](#) for details.
Index Refer to [QEStringFormatting::INDEX](#) for details.

9.96.2.2 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.

9.96.2.3 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.

9.96.2.4 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.

9.96.2.5 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.

9.96.2.6 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.

9.96.3 Constructor & Destructor Documentation

9.96.3.1 QELabel::QELabel (`QWidget * parent = 0`)

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

9.96.3.2 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.

9.96.4 Member Function Documentation

9.96.4.1 void QELabel::dbValueChanged (`const QString & out`) [signal]

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

9.96.5 Property Documentation

9.96.5.1 bool QELabel::addUnits [read, write]

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

9.96.5.2 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](#).

9.96.5.3 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'.

9.96.5.4 bool QELabel::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [standardProperties](#).

9.96.5.5 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).

9.96.5.6 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.

9.96.5.7 bool QELabel::leadingZero [read, write]

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

9.96.5.8 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)
 >= Greater 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"

9.96.5.9 Notations QELabel::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.96.5.10 QPixmap QELabel::pixmap0 [read, write]

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

9.96.5.11 QPixmap QELabel::pixmap1 [read, write]

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

9.96.5.12 QPixmap QELabel:: pixmap2 [read, write]

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

9.96.5.13 QPixmap QELabel:: pixmap3 [read, write]

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

9.96.5.14 QPixmap QELabel:: pixmap4 [read, write]

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

9.96.5.15 QPixmap QELabel:: pixmap5 [read, write]

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

9.96.5.16 QPixmap QELabel:: pixmap6 [read, write]

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

9.96.5.17 QPixmap QELabel:: pixmap7 [read, write]

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

9.96.5.18 int QELabel:: precision [read, write]

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

9.96.5.19 bool QELabel:: trailingZeros [read, write]

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

9.96.5.20 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.

9.96.5.21 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.

9.96.5.22 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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.96.5.23 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 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.

9.96.5.24 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 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.

9.96.5.25 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.

9.96.5.26 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 programmatically 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'.

9.96.5.27 QString QELabel::variable [read, write]

EPICS variable name (CA PV)

9.96.5.28 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](#).

9.96.5.29 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 also used for other purposes.

9.96.5.30 bool QELabel::visible [read, write]

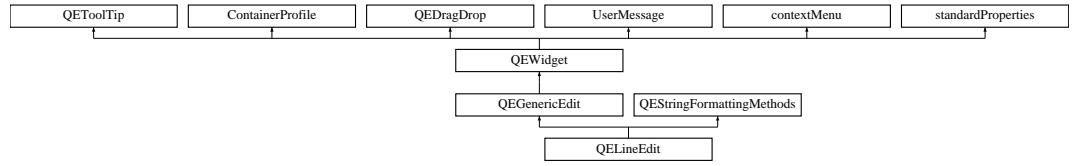
Display the widget. Default is true. Setting this property false is useful if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QELabel/QELabel.h
- /tmp/epicsqt/trunk/framework/widgets/QELabel/QELabel.cpp

9.97 QELineEdit Class Reference

Inheritance diagram for QELineEdit:



Public Types

- 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` }

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

- 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.
- [QELlineEdit \(QWidget *parent=0\)](#)
- [QELlineEdit \(const QString &variableName, QWidget *parent=0\)](#)

Properties

- int [precision](#)
- bool [useDbPrecision](#)
- bool [leadingZero](#)
- bool [trailingZeros](#)
- bool [addUnits](#)
- QString [localEnumeration](#)
- [Formats format](#)
- unsigned [int](#)
- [Notations notation](#)
- [ArrayActions arrayAction](#)

9.97.1 Member Enumeration Documentation

9.97.1.1 enum QELlineEdit::ArrayActions

User friendly enumerations for arrayAction property - refer to [QStringFormatting::arrayActions](#) for details.

Enumerator:

- Append** Refer to [QStringFormatting::APPEND](#) for details.
Ascii Refer to [QStringFormatting::ASCII](#) for details.
Index Refer to [QStringFormatting::INDEX](#) for details.

9.97.1.2 enum QELlineEdit::Formats

User friendly enumerations for format property - refer to [QStringFormatting::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.

9.97.1.3 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.

9.97.2 Constructor & Destructor Documentation

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

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

9.97.2.2 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.

9.97.3 Member Function Documentation

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

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

9.97.4 Property Documentation

9.97.4.1 bool QELineEdit::addUnits [read, write]

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

9.97.4.2 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'.

9.97.4.3 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).

9.97.4.4 unsigned QELineEdit::int [read, write]

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.

Reimplemented from [QEGenericEdit](#).

9.97.4.5 bool QELineEdit::leadingZero [read, write]

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

9.97.4.6 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)
>= Greater 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"

9.97.4.7 Notations QELineEdit::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.97.4.8 int QELineEdit::precision [read, write]

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

9.97.4.9 bool QELineEdit::trailingZeros [read, write]

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

9.97.4.10 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.

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

- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QELineEdit.h
- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QELineEdit.cpp

9.98 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

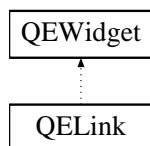
- `QString name () const`
- `QString toolTip () const`
- `QString whatsThis () const`
- `QWidget * createWidget (QWidget *parent)`
- `void initialize (QDesignerFormEditorInterface *core)`

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

- `/tmp/epicsqt/trunk/framework/widgets/QELineEdit/QELineEditManager.h`

9.99 QELink Class Reference

Inheritance diagram for QELink:



Public Types

- enum **conditions** {

CONDITION_EQ, **CONDITION_NE**, **CONDITION_GT**, **CONDITION_GE**,

CONDITION_LT, **CONDITION_LE** }
- enum **ConditionNames** {

Equal = QELink::CONDITION_EQ, **NotEqual** = QELink::CONDITION_NE, **GreaterThan**
 = QELink::CONDITION_GT, **GreaterThanOrEqual** = QELink::CONDITION_GE,

LessThan = QELink::CONDITION_LT, **LessThanOrEqual** = QELink::CONDITION_-
 LE }

Public Slots

- `void in (const bool &in)`
- `void in (const long &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 **setConditionProperty** (ConditionNames condition)
- ConditionNames **getConditionProperty** ()

Protected Attributes

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

Properties

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

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

- /tmp/epicsqt/trunk/framework/widgets/QELink/QELink.h
- /tmp/epicsqt/trunk/framework/widgets/QELink/QELink.cpp

9.100 QELocalEnumeration Class Reference

```
#include <QELocalEnumeration.h>
```

Classes

- class **localEnumerationItem**

Public Member Functions

- `QELocalEnumeration ()`
- `QELocalEnumeration (const QString &localEnumeration)`
- void `setLocalEnumeration (const QString &localEnumeration)`
- `QString getLocalEnumeration ()`
- bool `isDefined ()`
- `QString valueToText (const QVariant &value, bool &match)`
- `QVariant textToValue (const QString &text, bool &ok)`
- int `textToInt (const QString &text, bool &ok)`
- double `texttoDouble (const QString &text, bool &ok)`

9.100.1 Detailed Description

This class allows a user defined two-way value to enumeration map. The map is define using a single string, typically a widget property string. This may then be used to replace the enumeration values provided by EPICS and/or provide an enueration set of more that 16 values. See [setLocalEnumeration\(\)](#) for the use of 'localEnumeration'.

This functionality that this class provided was formerly embedded within [QEStringFormatting](#).

9.100.2 Constructor & Destructor Documentation

9.100.2.1 QELocalEnumeration::QELocalEnumeration ()

Constructors

9.100.2.2 QELocalEnumeration::QELocalEnumeration (const QString & *localEnumeration*)

Constructor with localEnumeration

9.100.3 Member Function Documentation

9.100.3.1 `QString QELocalEnumeration::getLocalEnumeration()`

Get the local enumeration strings. See [setLocalEnumeration\(\)](#) for the use of 'localEnumeration'.

9.100.3.2 `bool QELocalEnumeration::isDefined()`

Evaluates: `getLocalEnumeration.count() > 0`

9.100.3.3 `void QELocalEnumeration::setLocalEnumeration (const QString & localEnumeration)`

Parse the local enumeration string.

Format is:

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

Where: < Less than <= Less than or equal = Equal (default if no operator specified)
>= Greater 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"`

Will completely re-initialises the object.

9.100.3.4 `double QELocalEnumeration::text.ToDouble (const QString & text, bool & ok)`

Generate a double value given a string, using formatting defined within this class. If the value can be formatted the formatted value is returned and 'ok' is true. If the value can't

be formatted then 0.0 is returned and 'ok' is false.

9.100.3.5 int QELocalEnumeration::textToInt (const QString & *text*, bool & *ok*)

Generate an integer value given a string, using formatting defined within this class. If the value can be formatted the formatted value is returned and 'ok' is true. If the value can't be formatted then 0 is returned and 'ok' is false.

9.100.3.6 QVariant QELocalEnumeration::textToValue (const QString & *text*, bool & *ok*)

Generate a value given a string, using formatting defined within this class. If the value can be formatted the formatted value is returned and 'ok' is true. If the value can't be formatted an error string is returned and 'ok' is false

9.100.3.7 QString QELocalEnumeration::valueToString (const QVariant & *value*, bool & *match*)

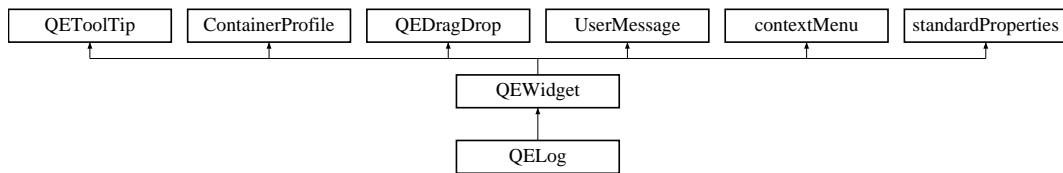
Format a variant value using local enumeration list. If the value is numeric, then the value is compared to the numeric interpretation of the enumeration values, if the value is textual, then the value is compared to the textual enumeration values.

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

- /tmp/epicsqt/trunk/framework/data/include/QELocalEnumeration.h
- /tmp/epicsqt/trunk/framework/data/src/QELocalEnumeration.cpp

9.101 QELog Class Reference

Inheritance diagram for QELog:



Public Types

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

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 **setOptionsLayout** (int pValue)
- int **getOptionsLayout** ()
- 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 **setOptionsLayoutProperty** (optionsLayoutProperty pOptionsLayout)
- optionsLayoutProperty **getOptionsLayoutProperty** ()
- MessageFilterOptions **getMessageFormFilter** ()
- void **setMessageFormFilter** (MessageFilterOptions messageFormFilter)
- MessageFilterOptions **getMessageSourceFilter** ()
- void **setMessageSourceFilter** (MessageFilterOptions messageSourceFilter)
- **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 Attributes

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

Properties

- `bool showColumnTime`
- `bool showColumnType`
- `bool showColumnMessage`
- `bool showMessageFilter`
- `bool showClear`
- `bool showSave`
- `optionsLayoutProperty optionsLayout`
- `QColor infoColor`
- `QColor warningColor`
- `QColor errorColor`
- `MessageFilterOptions messageFormFilter`
- `MessageFilterOptions messageSourceFilter`
- `unsigned int`
- `bool variableAsToolTip`
- `bool allowDrop`
- `bool visible`
- `QString userLevelUserStyle`
- `QString userLevelScientistStyle`
- `QString userLevelEngineerStyle`
- `UserLevels userLevelVisibility`
- `UserLevels userLevelEnabled`
- `bool displayAlarmState`

9.101.1 Member Enumeration Documentation

9.101.1.1 enum QELog::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.

9.101.2 Property Documentation

9.101.2.1 bool QELog::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](#).

9.101.2.2 bool QELog::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [StandardProperties](#).

9.101.2.3 unsigned QELog::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.

9.101.2.4 UserLevels QELog::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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.101.2.5 QString QELog::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.

9.101.2.6 `QString QELog::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.

9.101.2.7 `QString QELog::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.

9.101.2.8 `UserLevels QELog::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 programmatically 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'.

9.101.2.9 `bool QELog::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](#).

9.101.2.10 `bool QELog::visible [read, write]`

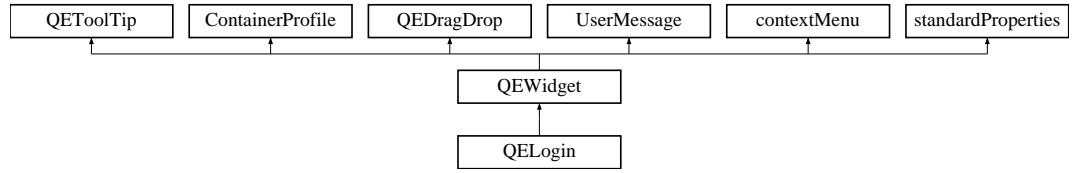
Display the widget. Default is true. Setting this property false is useful if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QELog/QELog.h
- /tmp/epicsqt/trunk/framework/widgets/QELog/QELog.cpp

9.102 QELogin Class Reference

Inheritance diagram for QELogin:



Signals

- void **login ()**

Public Member Functions

- **QELogin** (QWidget *pParent=0)
- bool **login** ([userLevelTypes::userLevels](#) level, QString password)
- QString **getPriorityUserPassword** ()
- QString **getPriorityScientistPassword** ()
- QString **getPriorityEngineerPassword** ()
- void **setUserPassword** (QString pValue)
- QString **getUserPassword** ()
- void **setScientistPassword** (QString pValue)
- QString **getScientistPassword** ()
- void **setEngineerPassword** (QString pValue)
- QString **getEngineerPassword** ()
- void **setCompactStyle** (bool compactStyle)
- bool **getCompactStyle** ()
- void **setStatusOnly** (bool statusOnlyIn)
- bool **getStatusOnly** ()
- QString **getUserTypeName** ([userLevelTypes::userLevels](#) type)

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

- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.h
- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.cpp

9.103 QELoginDialog Class Reference

Public Member Functions

- **QELoginDialog** ([QELogin](#) *ownerIn)

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

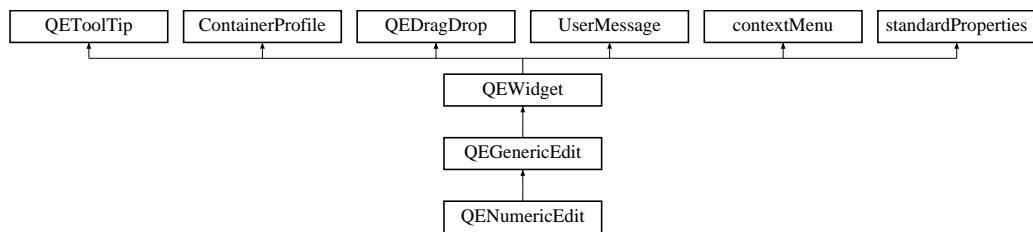
- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.h
- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.cpp

9.104 QENumericEdit Class Reference

The [QENumericEdit](#) class This class is similar to [QLineEdit](#) (both of which are derived from [QLineEdit](#)). However this class is tailored specifically for editing numerical values.

```
#include <QENumericEdit.h>
```

Inheritance diagram for QENumericEdit:



Signals

- void [dbValueChanged](#) (const double &out)

Public Member Functions

- [QENumericEdit](#) (QWidget *parent=0)
- [QENumericEdit](#) (const QString &variableName, QWidget *parent=0)
- virtual ~[QENumericEdit](#) ()
Destruction.
- double **getNumericValue** ()
- void **setAutoScale** (const bool value)
- bool **getAutoScale** ()
- void **setPropertyPrecision** (const int value)
- int **getPropertyPrecision** ()
- void **setPropertyLeadingZeros** (const int value)
- int **getPropertyLeadingZeros** ()
- void **setPropertyMinimum** (const double value)
- double **getPropertyMinimum** ()
- void **setPropertyMaximum** (const double value)
- double **getPropertyMaximum** ()
- void **setAddUnits** (bool addUnits)
- bool **getAddUnits** ()

- void **setRadix** (const QEFixedPointRadix::Radicies value)
- QEFixedPointRadix::Radicies **getRadix** ()
- void **setSeparator** (const QEFixedPointRadix::Separators value)
- QEFixedPointRadix::Separators **getSeparator** ()

Protected Member Functions

- void **keyPressEvent** (QKeyEvent *event)
- void **focusInEvent** (QFocusEvent *event)
- void **mouseReleaseEvent** (QMouseEvent *event)
- void **establishConnection** (unsigned int variableIndex)
- **qcaobject::QCaObject** * **createQcalItem** (unsigned int variableIndex)
- int **getPrecision** ()
- int **getLeadingZeros** ()
- double **getMinimum** ()
- double **getMaximum** ()
- int **maximumSignificance** ()
- int **getRadixValue** ()
- void **setValue** (const QVariant &value)

Sets the underlying QLineEdit widget to the given value.
- QVariant **getValue** ()

Gets the underlying value.
- bool **writeData** (const QVariant &value, QString &message)

Write the data to the channel.

Protected Attributes

- QEFloatingFormatting **floatingFormatting**

Properties

- bool **autoScale**
- QEFixedPointRadix::Radicies **radix**

Specify radix, default is Decimal.
- QEFixedPointRadix::Separators **separator**

Specify digit 'thousands' separator character, default is none.
- int **precision**
- int **leadingZeros**
- double **minimum**
- double **maximum**
- bool **addUnits**

Friends

- class **NumericValidator**

9.104.1 Detailed Description

The [QENumericEdit](#) class This class is similar to [QELineEdit](#) (both of which are derived from [QLineEdit](#)). However this class is tailored specifically for editing numerical values.

Note: this class based on thumb_wheel_edits.pas by same author.

9.104.2 Constructor & Destructor Documentation

9.104.2.1 `QENumericEdit::QENumericEdit (QWidget * parent = 0)`

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

9.104.2.2 `QENumericEdit::QENumericEdit (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.

9.104.3 Member Function Documentation

9.104.3.1 `void QENumericEdit::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.

9.104.4 Property Documentation

9.104.4.1 `bool QENumericEdit::addUnits [read, write]`

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

9.104.4.2 `bool QENumericEdit::autoScale [read, write]`

If true (default), display and editing of numbers using the precision, and control limits supplied with the data. If false, the precision, leadingZeros, minimum and maximum properties are used.

9.104.4.3 `int QENumericEdit::leadingZeros [read, write]`

Specifies the number of leading zeros. This is only used if autoScale is false. Strictly speaking, this should be an unsigned int, but designer properties editor much 'nicer'

with integers.

9.104.4.4 double QENumericEdit::maximum [read, write]

Specifies the maximum allowed value. This is only used if autoScale is false.

9.104.4.5 double QENumericEdit::minimum [read, write]

Specifies the minimum allowed value. This is only used if autoScale is false.

9.104.4.6 int QENumericEdit::precision [read, write]

Precision used for the display and editing of numbers. The default is 4. This is only used if autoScale is false. Strictly speaking, this should be an unsigned int, but designer properties editor much 'nicer' with integers.

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

- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QENumericEdit.h
- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QENumericEdit.cpp

9.105 QENumericEditManager Class Reference

Public Member Functions

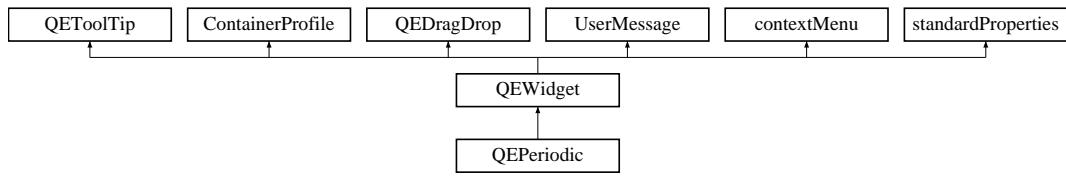
- **QENumericEditManager** (QObject *parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget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

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

- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QENumericEditManager.h
- /tmp/epicsqt/trunk/framework/widgets/QELineEdit/QENumericEditManager.cpp

9.106 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 = userLevelTypes::USERLEVEL_USER, Scientist = userLevelTypes::USERLEVEL_SCIENTIST, Engineer = userLevelTypes::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 }

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** ()
- **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 `mousePressEvent` (QMouseEvent *event)
- void `setDrop` (QVariant drop)
- QVariant `getDrop` ()
- QString `copyVariable` ()
- QVariant `copyData` ()
- void `paste` (QVariant s)

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 `visible`
- unsigned `int`
- QString `userLevelUserStyle`
- QString `userLevelScientistStyle`
- QString `userLevelEngineerStyle`
- UserLevels `userLevelVisibility`
- UserLevels `userLevelEnabled`
- bool `displayAlarmState`
- PresentationOptions `presentationOption`
- VariableTypes `variableType1`
- VariableTypes `variableType2`
- QString `userInfo`

9.106.1 Member Enumeration Documentation

9.106.1.1 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.

9.106.2 Member Function Documentation

9.106.2.1 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.

9.106.2.2 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.

9.106.3 Member Data Documentation

9.106.3.1 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](#).

9.106.4 Property Documentation

9.106.4.1 bool QEPeriodic::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [StandardProperties](#).

9.106.4.2 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.

9.106.4.3 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.

9.106.4.4 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.

9.106.4.5 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](#).

9.106.4.6 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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.106.4.7 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 `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.

9.106.4.8 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 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.

9.106.4.9 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.

9.106.4.10 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 programmatically 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'.

9.106.4.11 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 [QToolTip](#).

9.106.4.12 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.

9.106.4.13 bool QEPeriodic::visible [read, write]

Display the widget. Default is true. Setting this property false is useful 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.

9.106.4.14 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.

9.106.4.15 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:

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodic.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodic.cpp

9.107 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:

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodic.h

9.108 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:

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodicTaskMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodicTaskMenuExtension.cpp

9.109 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:

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodicTaskMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodicTaskMenuExtension.cpp

9.110 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 **valueInitiated** (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

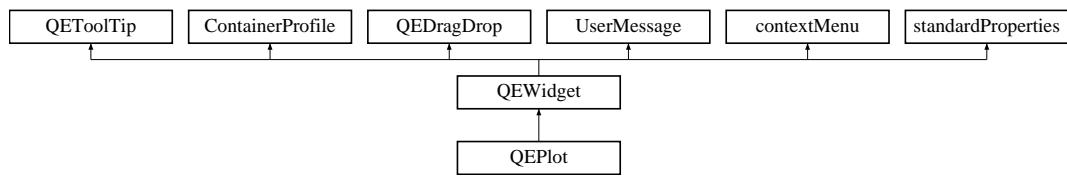
- static void **setDebugLevel** (unsigned level=0)
- static QVariant **get** (const QString &_pvName, int delay=defaultDelay)
- static QVariant **set** (QString &_pvName, const QVariant &value, int delay=-1)

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

- /tmp/epicsqt/trunk/framework/data/include/qepicspv.h
- /tmp/epicsqt/trunk/framework/data/src/qepicspv.cpp

9.111 QEPlot Class Reference

Inheritance diagram for QEPlot:



Public Types

- enum **UserLevels** { **User** = userLevelTypes::USERLEVEL_USER, **Scientist** = userLevelTypes::USERLEVEL_SCIENTIST, **Engineer** = userLevelTypes::USERLEVEL_ENGINEER }
- enum **TraceStyles** { **Lines** = QwtPlotCurve::Lines, **Sticks** = QwtPlotCurve::Sticks, **Steps** = QwtPlotCurve::Steps, **Dots** = QwtPlotCurve::Dots }

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** ()
- [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** ()
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)

Protected Attributes

- [QEFormatting floatingFormatting](#)
- bool **localEnabled**
- bool **allowDrop**

Properties

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

9.111.1 Member Enumeration Documentation

9.111.1.1 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.

9.111.2 Member Function Documentation

9.111.2.1 `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.

9.111.2.2 `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.

9.111.3 Member Data Documentation

9.111.3.1 `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](#).

9.111.4 Property Documentation

9.111.4.1 `bool QEPlot::displayAlarmState [read, write]`

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [standardProperties](#).

9.111.4.2 `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.

9.111.4.3 `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 programmatically through

`setUserLevel()` Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.111.4.4 `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 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.

9.111.4.5 `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 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.

9.111.4.6 `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.

9.111.4.7 `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 `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'.

9.111.4.8 `QString QEPlot::variable1` [read, write]

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

9.111.4.9 QString QEPlot::variable2 [read, write]

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

9.111.4.10 QString QEPlot::variable3 [read, write]

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

9.111.4.11 QString QEPlot::variable4 [read, write]

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

9.111.4.12 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](#).

9.111.4.13 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.

9.111.4.14 bool QEPlot::visible [read, write]

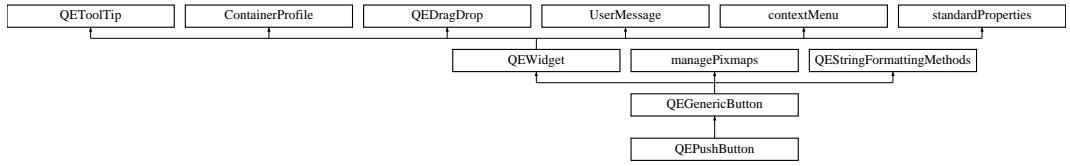
Display the widget. Default is true. Setting this property false is useful if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEPlot/QEPlot.h
- /tmp/epicsqt/trunk/framework/widgets/QEPlot/QEPlot.cpp

9.112 QEPushButton Class Reference

Inheritance diagram for QEPushButton:



Public Types

- enum `UserLevels` { `User` = userLevelTypes::USERLEVEL_USER, `Scientist` = userLevelTypes::USERLEVEL_SCIENTIST, `Engineer` = userLevelTypes::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` = QEGenericButton::UPDATE_TEXT, `Icon` = QEGenericButton::UPDATE_ICON, `TextAndIcon` = QEGenericButton::UPDATE_TEXT_AND_ICON, `State` = QEGenericButton::UPDATE_STATE }

User friendly enumerations for updateOption property - refer to QEGenericButton::updateOptions for details.
- enum `ProgramStartupOptionNames` { `None` = applicationLauncher::PSO_NONE, `Terminal` = applicationLauncher::PSO_TERMINAL, `LogOutput` = applicationLauncher::PSO_LOGOUTPUT, `StdOutput` = applicationLauncher::PSO_STDOUPUT }
- enum `CreationOptionNames` {

`Open` = QEActionRequests::OptionOpen, `NewTab` = QEActionRequests::OptionNewTab, `NewWindow` = QEActionRequests::OptionNewWindow, `DockTop` = QEActionRequests::OptionTopDockWindow,

`DockBottom` = QEActionRequests::OptionBottomDockWindow, `DockLeft` = QEActionRequests::OptionLeftDockWindow, `DockRight` = QEActionRequests::OptionRightDockWindow, `DockTopTabbed` = QEActionRequests::OptionTopDockWindowTabbed, `DockBottomTabbed` = QEActionRequests::OptionBottomDockWindowTabbed, `DockLeftTabbed` = QEActionRequests::OptionLeftDockWindowTabbed, `DockRightTabbed` = QEActionRequests::OptionRightDockWindowTabbed, `DockFloating` = QEActionRequests::OptionFloatingDockWindow }

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 `requestAction` (const QEActionRequests &request)

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` (const QEActionRequests &request)
Internal use only. Request a new GUI is created. Typically, this is caught by the QEgui application.
- void `pressed` (int value)
- void `released` (int value)
- void `clicked` (int value)
- void `programCompleted` ()
Program started by button has completed.

Public Member Functions

- `QEPushButton` (QWidget *parent=0)
- `QEPushButton` (const QString &variableName, QWidget *parent=0)
- `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 allowDrop`
- `bool visible`
- `unsigned int`
- `QString userLevelUserStyle`
- `QString userLevelScientistStyle`
- `QString userLevelEngineerStyle`
- `UserLevels userLevelVisibility`
- `UserLevels userLevelEnabled`
- `bool displayAlarmState`
- `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`
- `QString confirmText`
- `bool writeOnPress`
- `bool writeOnRelease`
- `bool writeOnClick`
- `QString pressText`
- `QString releaseText`
- `QString clickText`
- `QString clickCheckedText`
- `QString labelText`
- `QString program`

- QStringList [arguments](#)
- [ProgramStartupOptionNames](#) [programStartupOption](#)
- QString [guiFile](#)
- [CreationOptionNames](#) [creationOption](#)
- QString [prioritySubstitutions](#)
- QString [customisationName](#)

9.112.1 Member Enumeration Documentation

9.112.1.1 enum QEPushButton::ArrayActions

User friendly enumerations for arrayAction property - refer to [QEStringFormatting::arrayActions](#) for details.

Enumerator:

- Append** Refer to [QEStringFormatting::APPEND](#) for details.
Ascii Refer to [QEStringFormatting::ASCII](#) for details.
Index Refer to [QEStringFormatting::INDEX](#) for details.

9.112.1.2 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.
DockTop Open new GUI in a top dock window.
DockBottom Open new GUI in a bottom dock window.
DockLeft Open new GUI in a left dock window.
DockRight Open new GUI in a right dock window.
DockTopTabbed Open new GUI in a top dock window (tabbed with any existing dock in that area)
DockBottomTabbed Open new GUI in a bottom dock window (tabbed with any existing dock in that area)
DockLeftTabbed Open new GUI in a left dock window (tabbed with any existing dock in that area)
DockRightTabbed Open new GUI in a right dock window (tabbed with any existing dock in that area)
DockFloating Open new GUI in a floating dock window.

9.112.1.3 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.

9.112.1.4 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.

9.112.1.5 enum QEPushButton::ProgramStartupOptionNames

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

Enumerator:

None Just run the program.

Terminal Run the program in a terminal (in Windows a command interpreter will also be started, so the program may be a built-in command like 'dir')

LogOutput Run the program, and log the output in the QE message system.

StdOutput Run the program, and send output to standard output and standard error.

9.112.1.6 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)

9.112.1.7 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.

9.112.2 Constructor & Destructor Documentation**9.112.2.1 QEPushButton::QEPushButton (QWidget * *parent* = 0)**

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

9.112.2.2 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.

9.112.3 Member Function Documentation**9.112.3.1 void QEPushButton::clicked (int *value*) [signal]**

Button has been Clicked. The value emitted is the integer interpretation of the [clickText](#) property (or the [clickCheckedText](#) property if the button was checked)

9.112.3.2 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.

9.112.3.3 void QEPushButton::pressed (int value) [signal]

Button has been Pressed. The value emitted is the integer interpretation of the press-Text property

9.112.3.4 void QEPushButton::released (int value) [signal]

Button has been Released The value emitted is the integer interpretation of the release-Text property

9.112.3.5 void QEPushButton::requestAction (const QEActionRequests & request) [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.

9.112.4 Property Documentation

9.112.4.1 bool QEPushButton::addUnits [read, write]

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

9.112.4.2 Qt::Alignment QEPushButton::alignment [read, write]

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

9.112.4.3 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](#).

9.112.4.4 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.

9.112.4.5 QStringList QEPushButton::arguments [read, write]

Arguments for program specified in the 'program' property.

9.112.4.6 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'.

9.112.4.7 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 display 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 display 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](#).

9.112.4.8 QString QEPushButton::clickText [read, write]

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

Reimplemented from [QEGenericButton](#).

9.112.4.9 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

9.112.4.10 `QString QEPushButton::confirmText` [read, write]

Text used to confirm action if confirmation dialog is presented

Reimplemented from [QEGenericButton](#).

9.112.4.11 `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](#).

9.112.4.12 `QString QEPushButton::customisationName` [read, write]

Window customisation name. This name will be used to select a set of window customisations including menu items and tool bar buttons. Applications such as QEGui can load .xml files containing named sets of window customisations. This property is used to select a set loaded from these files. The selected set of customisations will be applied to the main window containing the new GUI. Customisations are not applied if the GUI is opened as a dock.

Reimplemented from [QEGenericButton](#).

9.112.4.13 `bool QEPushButton::displayAlarmState` [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [standardProperties](#).

9.112.4.14 `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).

9.112.4.15 `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::openQEFfile\(\)](#) in QEWidget.cpp for details.

9.112.4.16 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.

9.112.4.17 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. IF NOT LEFT EMPTY, THIS TEXT WILL TAKE PRIORITY OVER THE PUSH BUTTON 'text' PROPERTY! 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](#).

9.112.4.18 bool QEPushButton::leadingZero [read, write]

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

9.112.4.19 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)
=> Greater 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"

9.112.4.20 Notations QEPushButton::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.112.4.21 QString QEPushButton::password [read, write]

Password user will need to enter before any action is taken

Reimplemented from [QEGenericButton](#).

9.112.4.22 QPixmap QEPushButton:: pixmap0 [read, write]

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

9.112.4.23 QPixmap QEPushButton:: pixmap1 [read, write]

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

9.112.4.24 QPixmap QEPushButton:: pixmap2 [read, write]

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

9.112.4.25 QPixmap QEPushButton:: pixmap3 [read, write]

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

9.112.4.26 QPixmap QEPushButton:: pixmap4 [read, write]

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

9.112.4.27 QPixmap QEPushButton:: pixmap5 [read, write]

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

9.112.4.28 QPixmap QEPushButton:: pixmap6 [read, write]

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

9.112.4.29 QPixmap QEPushButton:: pixmap7 [read, write]

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

9.112.4.30 int QEPushButton:: precision [read, write]

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

9.112.4.31 QString QEPushButton:: pressText [read, write]

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

Reimplemented from [QEGenericButton](#).

9.112.4.32 QString QEPushButton:: prioritySubstitutions [read, write]

Overriding macro substitutions. These macro substitutions take precedence over any existing macro substitutions defined by the variableSubstitutions property, any parent forms, or the application containing the button. These macro substitutions are particularly useful when the button's function is to reload the same form but with different macro substitutions. The variableSubstitutions property cannot be used for this since, although they are added to the list of macro substitutions applied to the new form, they are appended to the list and the existing macro substitutions take precedence.

Reimplemented from [QEGenericButton](#).

9.112.4.33 `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

9.112.4.34 `ProgramStartupOptionNames QEPushButton::programStartupOption`
[read, write]

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

9.112.4.35 `QString QEPushButton::releaseText` [read, write]

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

Reimplemented from [QEGenericButton](#).

9.112.4.36 `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](#).

9.112.4.37 `bool QEPushButton::trailingZeros` [read, write]

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

9.112.4.38 `UpdateOptions QEPushButton::updateOption` [read, write]

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

Reimplemented from [QEGenericButton](#).

9.112.4.39 `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.

9.112.4.40 `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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets

that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.112.4.41 `QString QEPushbutton::userLevelEngineerStyle [read, write]`

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

9.112.4.42 `QString QEPushbutton::userLevelScientistStyle [read, write]`

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

9.112.4.43 `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.

9.112.4.44 `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 programmatically 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'.

9.112.4.45 `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.

9.112.4.46 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](#).

9.112.4.47 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 also used for other purposes.

9.112.4.48 bool QEPushButton::visible [read, write]

Display the widget. Default is true. Setting this property false is useful 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.

9.112.4.49 bool QEPushButton::writeOnClick [read, write]

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

Reimplemented from [QEGenericButton](#).

9.112.4.50 bool QEPushButton::writeOnPress [read, write]

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

Reimplemented from [QEGenericButton](#).

9.112.4.51 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:

- /tmp/epicsqt/trunk/framework/widgets/QEButton/QEPushButton.h
- /tmp/epicsqt/trunk/framework/widgets/QEButton/QEPushButton.cpp

9.113 QEPVNameLists Class Reference

Public Member Functions

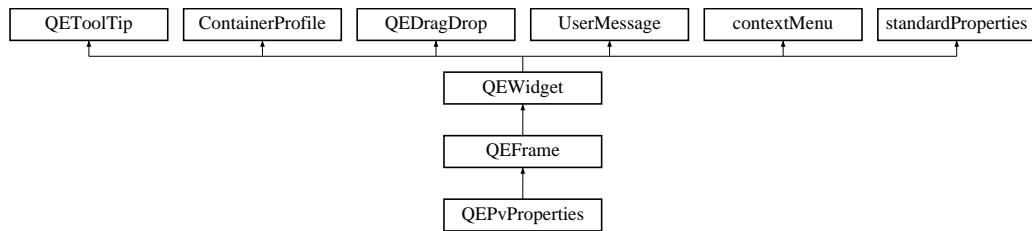
- void **prependOrMoveToFirst** (const QString &item)
- void **saveConfiguration** ([PMElement](#) &parentElement)
- void **restoreConfiguration** ([PMElement](#) &parentElement)

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

• /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChart.cpp

9.114 QEPvProperties Class Reference

Inheritance diagram for QEPvProperties:



Signals

- void **setCurrentBoxIndex** (int index)

Public Member Functions

- **QEPvProperties** (QWidget *parent=0)
- **QEPvProperties** (const QString &variableName, QWidget *parent=0)
- QSize **sizeHint** () const

Protected Member Functions

- void **resizeEvent** (QResizeEvent *event)
- void **establishConnection** (unsigned int variableIndex)
- [qcaobject::QCaObject](#) * **createQcalItem** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent *event)
- void **dropEvent** (QDropEvent *event)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()
- void **saveConfiguration** ([PersistanceManager](#) *pm)
- void **restoreConfiguration** ([PersistanceManager](#) *pm, [restorePhases](#) restorePhase)

- `QString copyVariable ()`
- `QVariant copyData ()`
- `void paste (QVariant s)`

Properties

- `QString variable`
- `QString variableSubstitutions`

9.114.1 Member Function Documentation

**9.114.1.1 void QEPvProperties::restoreConfiguration (`PersistanceManager *`,
`restorePhases`) [protected, virtual]**

Service a request to restore the QE widget's configuration. A QE widget recover any configuration details from the `PersistanceManager`. For example, a `QEStripChart` may restore the variables being plotted. Many QE widgets do not have any persistant data requirements and do not implement this method. This is called twice with an incrementing restorePhase. Most widgets will miss the first call as they don't exist yet (they are created as part of the first phase)

Reimplemented from `QEWidget`.

**9.114.1.2 void QEPvProperties::saveConfiguration (`PersistanceManager *`)
[protected, virtual]**

Service a request to save the QE widget's current configuration. A widget may save any configuration details through the `PersistanceManager`. For example, a `QEStripChart` may save the variables being plotted. Many QE widgets do not have any persistant data requirements and do not implement this method.

Reimplemented from `QEWidget`.

9.114.2 Property Documentation

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

EPICS variable name (CA PV)

9.114.2.2 `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.

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvProperties.h
- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvProperties.cpp

9.115 QEPvPropertiesManager Class Reference

Public Member Functions

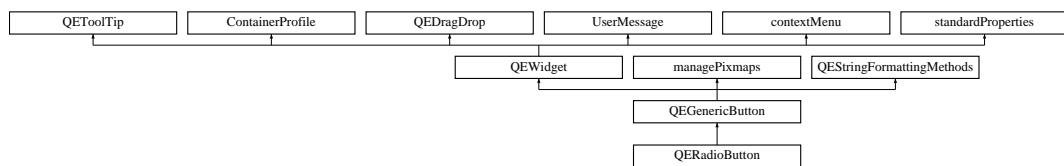
- **QEPvPropertiesManager** (QObject *parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget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

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesManager.h
- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesManager.cpp

9.116 QERadioButton Class Reference

Inheritance diagram for QERadioButton:



Public Types

- enum **UserLevels** { **User** = userLevelTypes::USERLEVEL_USER, **Scientist** = userLevelTypes::USERLEVEL_SCIENTIST, **Engineer** = userLevelTypes::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` = QEGenericButton::UPDATE_TEXT, `Icon` = QEGenericButton::UPDATE_ICON, `TextAndIcon` = QEGenericButton::UPDATE_TEXT_AND_ICON, `State` = QEGenericButton::UPDATE_STATE }

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

- enum `ProgramStartupOptionNames` { `None` = applicationLauncher::PSO_NONE, `Terminal` = applicationLauncher::PSO_TERMINAL, `LogOutput` = applicationLauncher::PSO_LOGOUTPUT, `StdOutput` = applicationLauncher::PSO_STDOUTPUT }
- enum `CreationOptionNames` {

`Open` = QEActionRequests::OptionOpen, `NewTab` = QEActionRequests::OptionNewTab,

`NewWindow` = QEActionRequests::OptionNewWindow, `DockTop` = QEActionRequests::OptionTopDockWindow,

`DockBottom` = QEActionRequests::OptionBottomDockWindow, `DockLeft` = QEActionRequests::OptionLeftDockWindow, `DockRight` = QEActionRequests::OptionRightDockWindow,

`DockTopTabbed` = QEActionRequests::OptionTopDockWindowTabbed,

`DockBottomTabbed` = QEActionRequests::OptionBottomDockWindowTabbed, `DockLeftTabbed` = QEActionRequests::OptionLeftDockWindowTabbed, `DockRightTabbed` = QEActionRequests::OptionRightDockWindowTabbed, `DockFloating` = QEActionRequests::OptionFloatingDockWindow
 }

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 `requestAction` (const QEActionRequests &request)

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` (const QEActionRequests &request)

Internal use only. Request a new GUI is created. Typically, this is caught by the QEGui application.
- void `pressed` (int value)
- void `released` (int value)
- void `clicked` (int value)
- void `programCompleted` ()

Program started by button has completed.

Public Member Functions

- `QERadioButton (QWidget *parent=0)`
- `QERadioButton (const QString &variableName, QWidget *parent=0)`
- `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 allowDrop`
- `bool visible`
- `unsigned int`
- `QString userLevelUserStyle`
- `QString userLevelScientistStyle`
- `QString userLevelEngineerStyle`
- `UserLevels userLevelVisibility`
- `UserLevels userLevelEnabled`
- `bool displayAlarmState`
- `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](#)
- QString [confirmText](#)
- bool [writeOnPress](#)
- bool [writeOnRelease](#)
- bool [writeOnClick](#)
- QString [pressText](#)
- QString [releaseText](#)
- QString [clickText](#)
- QString [clickCheckedText](#)
- QString [labelText](#)
- QString [program](#)
- QStringList [arguments](#)
- [ProgramStartupOptionNames](#) [programStartupOption](#)
- QString [guiFile](#)
- [CreationOptionNames](#) [creationOption](#)
- QString [prioritySubstitutions](#)
- QString [customisationName](#)

9.116.1 Member Enumeration Documentation

9.116.1.1 enum QERadioButton::ArrayActions

User friendly enumerations for arrayAction property - refer to [QEStringFormatting::arrayActions](#) for details.

Enumerator:

Append Refer to [QEStringFormatting::APPEND](#) for details.

Ascii Refer to [QEStringFormatting::ASCII](#) for details.

Index Refer to [QEStringFormatting::INDEX](#) for details.

9.116.1.2 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.
- DockTop*** Open new GUI in a top dock window.
- DockBottom*** Open new GUI in a bottom dock window.
- DockLeft*** Open new GUI in a left dock window.
- DockRight*** Open new GUI in a right dock window.
- DockTopTabbed*** Open new GUI in a top dock window (tabbed with any existing dock in that area)
- DockBottomTabbed*** Open new GUI in a bottom dock window (tabbed with any existing dock in that area)
- DockLeftTabbed*** Open new GUI in a left dock window (tabbed with any existing dock in that area)
- DockRightTabbed*** Open new GUI in a right dock window (tabbed with any existing dock in that area)
- DockFloating*** Open new GUI in a floating dock window.

9.116.1.3 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.

9.116.1.4 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.

9.116.1.5 enum QERadioButton::ProgramStartupOptionNames

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

Enumerator:

None Just run the program.

Terminal Run the program in a terminal (in Windows a command interpreter will also be started, so the program may be a built-in command like 'dir')

LogOutput Run the program, and log the output in the QE message system.

StdOutput Run the program, and send output to standard output and standard error.

9.116.1.6 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)

9.116.1.7 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.

9.116.2 Constructor & Destructor Documentation

9.116.2.1 `QERadioButton::QERadioButton (QWidget * parent = 0)`

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

9.116.2.2 `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.

9.116.3 Member Function Documentation

9.116.3.1 `void QERadioButton::clicked (int value) [signal]`

Button has been Clicked. The value emitted is the integer interpretation of the `clickText` property (or the `clickCheckedText` property if the button was checked)

9.116.3.2 `void QERadioButton::dbValueChanged (const QString & out) [signal]`

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

9.116.3.3 `void QERadioButton::pressed (int value) [signal]`

Button has been Pressed. The value emitted is the integer interpretation of the `pressText` property

9.116.3.4 `void QERadioButton::released (int value) [signal]`

Button has been Released The value emitted is the integer interpretation of the `releaseText` property

9.116.3.5 `void QERadioButton::requestAction (const QEActionRequests & request) [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.

9.116.4 Property Documentation

9.116.4.1 bool **QERadioButton::addUnits** [read, write]

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

9.116.4.2 Qt::Alignment **QERadioButton::alignment** [read, write]

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

9.116.4.3 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](#).

9.116.4.4 QStringList **QERadioButton::arguments** [read, write]

Arguments for program specified in the 'program' property.

9.116.4.5 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'.

9.116.4.6 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 display 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 display 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](#).

9.116.4.7 QString QERadioButton::clickText [read, write]

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

Reimplemented from [QEGenericButton](#).

9.116.4.8 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

9.116.4.9 QString QERadioButton::confirmText [read, write]

Text used to confirm action if confirmation dialog is presented

Reimplemented from [QEGenericButton](#).

9.116.4.10 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](#).

9.116.4.11 QString QERadioButton::customisationName [read, write]

Window customisation name. This name will be used to select a set of window customisations including menu items and tool bar buttons. Applications such as QEgui

can load .xml files containing named sets of window customisations. This property is used to select a set loaded from these files. The selected set of customisations will be applied to the main window containing the new GUI.

Reimplemented from [QEGenericButton](#).

9.116.4.12 bool QERadioButton::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [standardProperties](#).

9.116.4.13 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).

9.116.4.14 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.

9.116.4.15 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.

9.116.4.16 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. IF NOT LEFT EMPTY, THIS TEXT WILL TAKE PRIORITY OVER THE PUSH BUTTON 'text' PROPERTY! 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](#).

9.116.4.17 bool QERadioButton::leadingZero [read, write]

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

9.116.4.18 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)
>= Greater 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"

9.116.4.19 Notations QERadioButton::notation [read, write]

Notation used for numerical formatting. Default is fixed.

9.116.4.20 QString QERadioButton::password [read, write]

Password user will need to enter before any action is taken

Reimplemented from [QEGenericButton](#).

9.116.4.21 QPixmap QERadioButton:: pixmap0 [read, write]

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

9.116.4.22 QPixmap QERadioButton:: pixmap1 [read, write]

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

9.116.4.23 QPixmap QERadioButton:: pixmap2 [read, write]

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

9.116.4.24 QPixmap QERadioButton:: pixmap3 [read, write]

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

9.116.4.25 QPixmap QERadioButton:: pixmap4 [read, write]

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

9.116.4.26 QPixmap QERadioButton:: pixmap5 [read, write]

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

9.116.4.27 QPixmap QERadioButton:: pixmap6 [read, write]

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

9.116.4.28 QPixmap QERadioButton:: pixmap7 [read, write]

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

9.116.4.29 int QERadioButton::precision [read, write]

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

9.116.4.30 QString QERadioButton::pressText [read, write]

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

Reimplemented from [QEGenericButton](#).

9.116.4.31 QString QERadioButton::prioritySubstitutions [read, write]

Overriding macro substitutions. These macro substitions take precedence over any existing macro substitutions defined by the variableSubstitutions property, any parent forms, or the application containing the button. These macro substitutions are particularly usefull when the button's function is to reload the same form but with different macro substitutions. The variableSubstitutions property cannot be used for this since, although they are added to the list of macro substitions applied to the new form, they are appended to the list and the existing macro substitutions take precedence.

Reimplemented from [QEGenericButton](#).

9.116.4.32 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

9.116.4.33 ProgramStartupOptionNames QERadioButton::programStartupOption [read, write]

Startup options. Just run the command, run the command within a terminal, or display the output in QE message system.

9.116.4.34 QString QERadioButton::releaseText [read, write]

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

Reimplemented from [QEGenericButton](#).

9.116.4.35 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](#).

9.116.4.36 bool QERadioButton::trailingZeros [read, write]

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

9.116.4.37 UpdateOptions QERadioButton::updateOption [read, write]

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

Reimplemented from [QEGenericButton](#).

9.116.4.38 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.

9.116.4.39 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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.116.4.40 QString QERadioButton::userLevelEngineerStyle [read, write]

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

9.116.4.41 QString QERadioButton::userLevelScientistStyle [read, write]

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

9.116.4.42 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.

9.116.4.43 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 programmatically 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'.

9.116.4.44 QString QERadioButton::variable [read, write]

EPICS variable name (CA PV)

9.116.4.45 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](#).

9.116.4.46 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 also used for other purposes.

9.116.4.47 bool QERadioButton::visible [read, write]

Display the widget. Default is true. Setting this property false is useful 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.

9.116.4.48 bool QERadioButton::writeOnClick [read, write]

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

Reimplemented from [QEGenericButton](#).

9.116.4.49 bool QERadioButton::writeOnPress [read, write]

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

Reimplemented from [QEGenericButton](#).

9.116.4.50 bool QERadioButton::writeOnRelease [read, write]

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

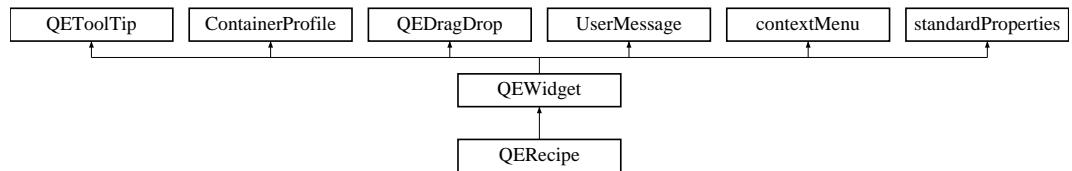
Reimplemented from [QEGenericButton](#).

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

- /tmp/epicsqt/trunk/framework/widgets/QEButton/QERadioButton.h
- /tmp/epicsqt/trunk/framework/widgets/QEButton/QERadioButton.cpp

9.117 QEReipe Class Reference

Inheritance diagram for QEReipe:



Public Types

- enum **configurationTypesProperty** { **File** = FROM_FILE, **Text** = FROM_TEXT }
- enum **optionsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **userTypesProperty** { **User** = userLevelTypes::USERLEVEL_USER, **Scientist** = userLevelTypes::USERLEVEL_SCIENTIST, **Engineer** = userLevelTypes::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 **setOptionsLayout** (int pValue)
- int **getOptionsLayout ()**
- void **setCurrentUserType** (int pValue)
- int **getCurrentUserType ()**
- bool **saveRecipeList ()**
- void **refreshRecipeList ()**
- void **refreshButton ()**
- void **userLevelChanged** ([userLevelTypes::userLevels](#) pValue)
- void **setConfigurationTypeProperty** (configurationTypesProperty pConfigurationType)
 - configurationTypesProperty **getConfigurationTypeProperty ()**
 - void **setOptionsLayoutProperty** (optionsLayoutProperty pOptionsLayout)
 - optionsLayoutProperty **getOptionsLayoutProperty ()**
 - 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 optionsLayout`
- `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`
- `optionsLayoutProperty optionsLayout`
- `userTypesProperty currentUserType`

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

- `/tmp/epicsqt/trunk/framework/widgets/QEReipe/QEReipe.h`
- `/tmp/epicsqt/trunk/framework/widgets/QEReipe/QEReipe.cpp`

9.118 QERecordFieldName Class Reference

Static Public Member Functions

- `static QString recordName (const QString &pvName)`
- `static QString fieldName (const QString &pvName)`
- `static QString fieldPvName (const QString &pvName, const QString &field)`
- `static QString rtypePvName (const QString &pvName)`
- `static bool pvNamesValid (const QString &pvName)`
- `static bool extractPvName (const QString &item, QString &pvName)`

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

- `/tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesUtilities.h`
- `/tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesUtilities.cpp`

9.119 QERecordSpec Class Reference

Public Member Functions

- **QERecordSpec** (const QString recordType)
- **QString getRecordType ()**
- **QString getFieldName** (const int index)

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesUtilities.h
- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesUtilities.cpp

9.120 QERecordSpecList Class Reference

Public Member Functions

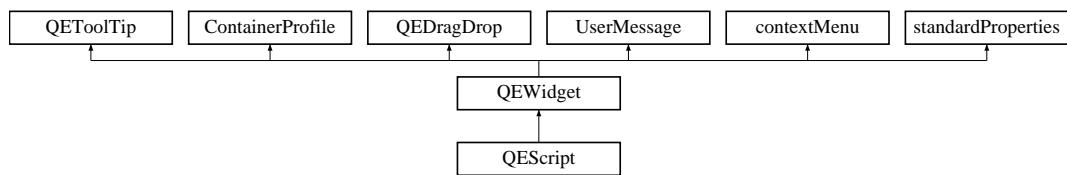
- **QERecordSpec * find** (const QString recordType)
- **void appendOrReplace** (QERecordSpec *recordSpec)
- **bool processRecordSpecFile** (const QString &filename)

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesUtilities.h
- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvPropertiesUtilities.cpp

9.121 QEScript Class Reference

Inheritance diagram for QEScript:



Public Types

- enum **optionsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **UserLevels** { **User** = userLevelTypes::USERLEVEL_USER, **Scientist** = userLevelTypes::USERLEVEL_SCIENTIST, **Engineer** = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void **selected** (QString pFilename)

Public Member Functions

- **QEScript** (QWidget *pParent=0)
- void **setShowScriptList** (bool pValue)
- bool **getShowScriptList** ()
- void **setShowNew** (bool pValue)
- bool **getShowNew** ()
- void **setShowSave** (bool pValue)
- bool **getShowSave** ()
- void **setShowDelete** (bool pValue)
- bool **getShowDelete** ()
- void **setShowExecute** (bool pValue)
- bool **getShowExecute** ()
- void **setShowTable** (bool pValue)
- bool **getShowTable** ()
- void **setShowTableControl** (bool pValue)
- bool **getShowTableControl** ()
- void **setShowColumnNumber** (bool pValue)
- bool **getShowColumnNumber** ()
- void **setShowColumnEnable** (bool pValue)
- bool **getShowColumnEnable** ()
- void **setShowColumnProgram** (bool pValue)
- bool **getShowColumnProgram** ()
- void **setShowColumnParameters** (bool pValue)
- bool **getShowColumnParameters** ()
- void **setShowColumnTimeOut** (bool pValue)
- bool **getShowColumnTimeOut** ()
- void **setShowColumnStop** (bool pValue)
- bool **getShowColumnStop** ()
- void **setShowColumnLog** (bool pValue)
- bool **getShowColumnLog** ()
- void **setScriptFile** (QString pValue)
- QString **getScriptFile** ()
- void **setExecuteText** (QString pValue)
- QString **getExecuteText** ()
- void **setOptionsLayout** (int pValue)
- int **getOptionsLayout** ()
- void **refreshScriptList** ()
- void **updateWidgets** ()
- void **setOptionsLayoutProperty** (optionsLayoutProperty pOptionsLayout)
- optionsLayoutProperty **getOptionsLayoutProperty** ()
- 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 Attributes

- QComboBox * **qComboBoxScriptList**
- QPushButton * **qPushButtonNew**
- QPushButton * **qPushButtonSave**
- QPushButton * **qPushButtonDelete**
- QPushButton * **qPushButtonExecute**
- QPushButton * **qPushButtonAdd**
- QPushButton * **qPushButtonRemove**
- QPushButton * **qPushButtonUp**
- QPushButton * **qPushButtonDown**
- QPushButton * **qPushButtonCopy**
- QPushButton * **qPushButtonPaste**
- **_QTableWidgetScript** * **qTableWidgetScript**
- QString **scriptFile**
- int **optionsLayout**
- QDomDocument **document**
- QString **filename**
- QList< **_CopyPaste** * > **copyPasteList**
- bool **isExecuting**

Properties

- bool **showScriptList**
- bool **showNew**
- bool **showSave**
- bool **showDelete**
- bool **showExecute**
- bool **showTable**
- bool **showTableControl**
- bool **showColumnNumber**
- bool **showColumnEnable**
- bool **showColumnProgram**

- bool **showColumnParameters**
- bool **showColumnTimeOut**
- bool **showColumnStop**
- bool **showColumnLog**
- QString **executeText**
- optionsLayoutProperty **optionsLayout**
- bool **variableAsToolTip**
- bool **allowDrop**
- bool **visible**
- unsigned **int**
- QString **userLevelUserStyle**
- QString **userLevelScientistStyle**
- QString **userLevelEngineerStyle**
- **UserLevels userLevelVisibility**
- **UserLevels userLevelEnabled**
- bool **displayAlarmState**

9.121.1 Member Enumeration Documentation

9.121.1.1 enum QEScript::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.

9.121.2 Property Documentation

9.121.2.1 bool QEScript::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](#).

9.121.2.2 bool QEScript::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [StandardProperties](#).

9.121.2.3 unsigned QEScript::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.

9.121.2.4 UserLevels QEScript::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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.121.2.5 QString QEScript::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.

9.121.2.6 QString QEScript::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.

9.121.2.7 QString QEScript::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.

9.121.2.8 UserLevels QEScript::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'.

9.121.2.9 bool QEScript::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 [QToolTip](#).

9.121.2.10 bool QEScript::visible [read, write]

Display the widget. Default is true. Setting this property false is useful 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.

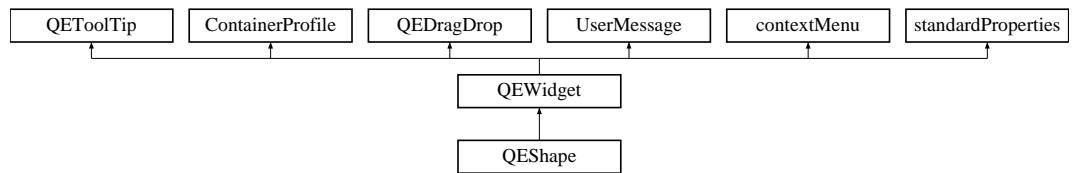
The documentation for this class was generated from the following files:

- /tmp/epicsqt/trunk/framework/widgets/QEScript/QEScript.h
 - /tmp/epicsqt/trunk/framework/widgets/QEScript/QEScript.cpp

9.122 QEShape Class Reference

```
#include <QEShape.h>
```

Inheritance diagram for QEShape:



Public Types

- enum **shapeOptions** {
 Line, Points, Polyline, Polygon,
 Rect, RoundedRectangle, Ellipse, Arc,
 Chord, Pie, Path }

- enum `animationOptions` {

`Width, Height, X, Y,`

`Transparency, Rotation, ColourHue, ColourSaturation,`

`ColourValue, ColourIndex, Penwidth }`
- enum `UserLevels` { `User` = userLevelTypes::USERLEVEL_USER, `Scientist` = userLevelTypes::USERLEVEL_SCIENTIST, `Engineer` = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void `dbValueChanged1` (const qulonglong &out)
- void `dbValueChanged2` (const qulonglong &out)
- void `dbValueChanged3` (const qulonglong &out)
- void `dbValueChanged4` (const qulonglong &out)
- void `dbValueChanged5` (const qulonglong &out)
- void `dbValueChanged6` (const qulonglong &out)

Public Member Functions

- `QEShape` (QWidget *parent=0)
- `QEShape` (const QString &variableName, QWidget *parent=0)
- void `scaleBy` (const int m, const int d)

Scale the widgets my m/d.
- 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.

- [`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 allowDrop`
- `bool visible`
- `unsigned int`
- `QString userLevelUserStyle`
- `QString userLevelScientistStyle`
- `QString userLevelEngineerStyle`
- `UserLevels userLevelVisibility`
- `UserLevels userLevelEnabled`
- `bool displayAlarmState`
- `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

9.122.1 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 ellipse can be drawn with four variables animating its vertical 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.

9.122.2 Member Enumeration Documentation

9.122.2.1 enum QEShape::animationOptions

Options for how a variable will animate the shape.

9.122.2.2 enum QEShape::shapeOptions

Options for the type of shape.

9.122.2.3 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.

9.122.3 Constructor & Destructor Documentation

9.122.3.1 QEShape::QEShape (`QWidget * parent = 0`)

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

9.122.3.2 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.

9.122.4 Member Function Documentation

9.122.4.1 void QEShape::dbValueChanged1 (`const qulonglong & 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.

9.122.4.2 void QEShape::dbValueChanged2 (`const qulonglong & 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.

9.122.4.3 void QEShape::dbValueChanged3 (`const qulonglong & 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.

9.122.4.4 void QEShape::dbValueChanged4 (const qulonglong & 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.

9.122.4.5 void QEShape::dbValueChanged5 (const qulonglong & 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.

9.122.4.6 void QEShape::dbValueChanged6 (const qulonglong & 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.

9.122.5 Property Documentation

9.122.5.1 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](#).

9.122.5.2 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.

9.122.5.3 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.

9.122.5.4 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.

9.122.5.5 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.

9.122.5.6 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.

9.122.5.7 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.

9.122.5.8 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'.

9.122.5.9 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'.

9.122.5.10 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'.

9.122.5.11 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'.

9.122.5.12 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'.

9.122.5.13 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'.

9.122.5.14 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'.

9.122.5.15 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'.

9.122.5.16 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'.

9.122.5.17 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'.

9.122.5.18 bool QEShape::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [StandardProperties](#).

9.122.5.19 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, RoundedRectangle, Ellipse, Arc, Chord, Pie, Path

9.122.5.20 double QEShape::offset1 [read, write]

Offset applied to data from the 1st variable before it is used to animate the shape

9.122.5.21 double QEShape::offset2 [read, write]

Offset applied to data from the 2nd variable before it is used to animate the shape

9.122.5.22 double QEShape::offset3 [read, write]

Offset applied to data from the 3rd variable before it is used to animate the shape

9.122.5.23 double QEShape::offset4 [read, write]

Offset applied to data from the 4th variable before it is used to animate the shape

9.122.5.24 double QEShape::offset5 [read, write]

Offset applied to data from the 5th variable before it is used to animate the shape

9.122.5.25 double QEShape::offset6 [read, write]

Offset applied to data from the 6th variable before it is used to animate the shape

9.122.5.26 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

9.122.5.27 QPoint QEShape::point10 [read, write]

10th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.122.5.28 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

9.122.5.29 QPoint QEShape::point3 [read, write]

3rd coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.122.5.30 QPoint QEShape::point4 [read, write]

4th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.122.5.31 QPoint QEShape::point5 [read, write]

5th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.122.5.32 QPoint QEShape::point6 [read, write]

6th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.122.5.33 QPoint QEShape::point7 [read, write]

7th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.122.5.34 QPoint QEShape::point8 [read, write]

8th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.122.5.35 QPoint QEShape::point9 [read, write]

9th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

9.122.5.36 double QEShape::scale2 [read, write]

Scale factor applied to data from the 2nd variable before it is used to animate the shape

9.122.5.37 double QEShape::scale3 [read, write]

Scale factor applied to data from the 3rd variable before it is used to animate the shape

9.122.5.38 double QEShape::scale4 [read, write]

Scale factor applied to data from the 4th variable before it is used to animate the shape

9.122.5.39 double QEShape::scale5 [read, write]

Scale factor applied to data from the 5th variable before it is used to animate the shape

9.122.5.40 double QEShape::scale6 [read, write]

Scale factor applied to data from the 6th variable before it is used to animate the shape

9.122.5.41 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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.122.5.42 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 `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.

9.122.5.43 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 `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.

9.122.5.44 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.

9.122.5.45 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 programmatically 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'.

9.122.5.46 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.

9.122.5.47 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.

9.122.5.48 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.

9.122.5.49 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.

9.122.5.50 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.

9.122.5.51 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.

9.122.5.52 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](#).

9.122.5.53 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.

9.122.5.54 bool QEShape::visible [read, write]

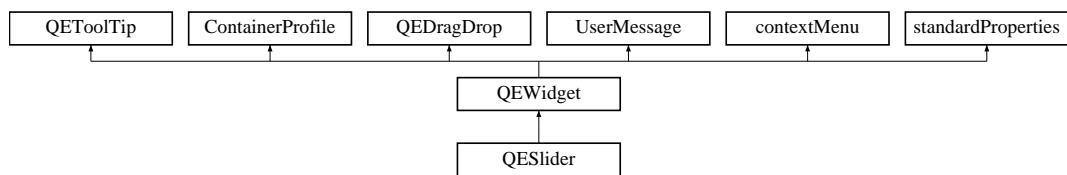
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QEShape/QEShape.h
- /tmp/epicsqt/trunk/framework/widgets/QEShape/QEShape.cpp

9.123 QESlider Class Reference

Inheritance diagram for QESlider:



Public Types

- enum [UserLevels](#) { [User](#) = userLevelTypes::USERLEVEL_USER, [Scientist](#) = userLevelTypes::USERLEVEL_SCIENTIST, [Engineer](#) = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void [dbValueChanged](#) (const qulonglong &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** ()
- 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** ()
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)

Protected Attributes

- **QEFloatingFormatting** `floatingFormatting`
- bool `writeOnChange`

Properties

- QString [variable](#)
- QString [variableSubstitutions](#)
- bool [subscribe](#)
- bool [variableAsToolTip](#)
- bool [allowDrop](#)
- bool [visible](#)
- unsigned [int](#)
- QString [userLevelUserStyle](#)
- QString [userLevelScientistStyle](#)
- QString [userLevelEngineerStyle](#)
- [UserLevels userLevelVisibility](#)
- [UserLevels userLevelEnabled](#)
- bool [displayAlarmState](#)

9.123.1 Member Enumeration Documentation

9.123.1.1 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.

9.123.2 Member Function Documentation

9.123.2.1 void QESlider::dbValueChanged (const qlonglong & out) [signal]

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

9.123.3 Member Data Documentation

9.123.3.1 bool QESlider::writeOnChange [read, write, protected]

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

9.123.4 Property Documentation

9.123.4.1 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](#).

9.123.4.2 bool QESlider::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [StandardProperties](#).

9.123.4.3 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.

9.123.4.4 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](#).

9.123.4.5 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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.123.4.6 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 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.

9.123.4.7 `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 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.

9.123.4.8 `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.

9.123.4.9 `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 programmatically 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'.

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

EPICS variable name (CA PV)

9.123.4.11 `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](#).

9.123.4.12 `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 also used for other purposes.

9.123.4.13 bool QESlider::visible [read, write]

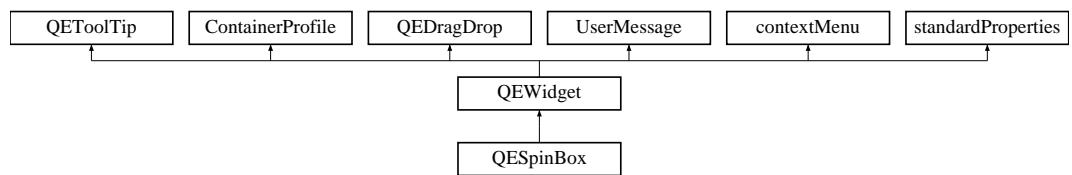
Display the widget. Default is true. Setting this property false is useful if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QESlider/QESlider.h
- /tmp/epicsqt/trunk/framework/widgets/QESlider/QESlider.cpp

9.124 QESpinBox Class Reference

Inheritance diagram for QESpinBox:



Public Types

- enum [UserLevels](#) { [User](#) = userLevelTypes::USERLEVEL_USER, [Scientist](#) = userLevelTypes::USERLEVEL_SCIENTIST, [Engineer](#) = userLevelTypes::USERLEVEL_ENGINEER }

Signals

- void [dbValueChanged](#) (const double &out)
- void [userChange](#) (const QString &oldValue, const QString &newValue, const QString &lastValue)

Internal use only. Used by [QEConfiguredLayout](#) to be notified when one of its widgets has written something.

Public Member Functions

- [QESpinBox](#) (QWidget *parent=0)
- [QESpinBox](#) (const QString &variableName, QWidget *parent=0)
- void [setWriteOnChange](#) (bool writeOnChangeIn)
- bool [getWriteOnChange](#) ()

- void **setSubscribe** (bool subscribe)
- bool **getSubscribe** ()
- void **setAddUnitsAsSuffix** (bool addUnitsAsSuffixIn)
- bool **getAddUnitsAsSuffix** ()
- void **setUseDbPrecisionForDecimals** (bool useDbPrecisionForDecimalIn)
- bool **getUseDbPrecisionForDecimals** ()
- **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** ()
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)
- QMenu * **getDefaultContextMenu** ()

Protected Attributes

- **QEFloatingFormatting floatingFormatting**
- bool **writeOnChange**
- bool **addUnitsAsSuffix**
- bool **useDbPrecisionForDecimal**

Properties

- QString **variable**
- QString **variableSubstitutions**
- bool **variableAsToolTip**
- bool **allowDrop**
- bool **visible**

- `unsigned int`
- `QString userLevelUserStyle`
- `QString userLevelScientistStyle`
- `QString userLevelEngineerStyle`
- `UserLevels userLevelVisibility`
- `UserLevels userLevelEnabled`
- `bool displayAlarmState`
- `bool subscribe`
- `bool useDbPrecision`
- `bool addUnits`

9.124.1 Member Enumeration Documentation

9.124.1.1 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.

9.124.2 Member Function Documentation

9.124.2.1 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.

9.124.3 Property Documentation

9.124.3.1 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](#).

9.124.3.2 bool QESpinBox::displayAlarmState [read, write]

If set (default) widget will indicate the alarm state of any variable data is displaying. Typically the background colour is set to indicate the alarm state. Note, this property is

included in the set of standard properties as it applies to most widgets. It will do nothing for widgets that don't display data.

Reimplemented from [standardProperties](#).

9.124.3.3 `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.

9.124.3.4 `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](#).

9.124.3.5 `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 programmatically through `setUserLevel()`. Widgets that are always accessible should be visible at 'User'. Widgets that are only accessible to scientists managing the facility should be visible at 'Scientist'. Widgets that are only accessible to engineers maintaining the facility should be visible at 'Engineer'.

9.124.3.6 `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.

9.124.3.7 `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.

9.124.3.8 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.

9.124.3.9 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 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'.

9.124.3.10 QString QESpinBox::variable [read, write]

EPICS variable name (CA PV)

9.124.3.11 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](#).

9.124.3.12 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 also used for other purposes.

9.124.3.13 bool QESpinBox::visible [read, write]

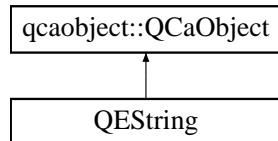
Display the widget. Default is true. Setting this property false is useful if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

- /tmp/epicsqt/trunk/framework/widgets/QESpinBox/QESpinBox.h
- /tmp/epicsqt/trunk/framework/widgets/QESpinBox/QESpinBox.cpp

9.125 QEString Class Reference

Inheritance diagram for QEString:



Public Slots

- void **writeString** (const QString &data)

Signals

- void **stringConnectionChanged** (QCaConnectionInfo &connectionInfo, const unsigned int &variableIndex)
- void **stringChanged** (const QString &value, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)

Public Member Functions

- **QEString** (QString recordName, QObject *eventObject, QEStringFormatting *stringFormattingIn, unsigned int variableIndexIn)
- **QEString** (QString recordName, QObject *eventObject, QEStringFormatting *stringFormattingIn, unsigned int variableIndexIn, UserMessage *userMessageIn)
- bool **writeString** (const QString &data, QString &message)

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

- /tmp/epicsqt/trunk/framework/data/include/QEString.h
- /tmp/epicsqt/trunk/framework/data/src/QEString.cpp

9.126 QEStringFormatting Class Reference

Public Types

- enum formats {

FORMAT_DEFAULT, FORMAT_FLOATING, FORMAT_INTEGER, FORMAT_UNSIGNEDINTEGER,

FORMAT_TIME, FORMAT_LOCAL_ENUMERATE, FORMAT_STRING }

- enum **notations** { **NOTATION_FIXED** = QTextStream::FixedNotation, **NOTATION_SCIENTIFIC** = QTextStream::ScientificNotation, **NOTATION_AUTOMATIC** = QTextStream::SmartNotation }
- enum **arrayActions** { **APPEND**, **ASCII**, **INDEX** }

Public Member Functions

- QString **formatString** (const QVariant &value)
- QVariant **formatValue** (const QString &text, bool &ok)
- void **setDbEgu** (QString egu)
- void **setDbEnumerations** (QStringList enumerations)
- void **setDbPrecision** (unsigned int dbPrecisionIn)
- void **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** ()

9.126.1 Member Enumeration Documentation

9.126.1.1 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.

9.126.1.2 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.

9.126.1.3 enum QEStringFormatting::notations

Notations when formatting a floating point number

Enumerator:

NOTATION_FIXED Standard floating point 123456.789.

NOTATION_SCIENTIFIC Scientific representation 1.23456789e6.

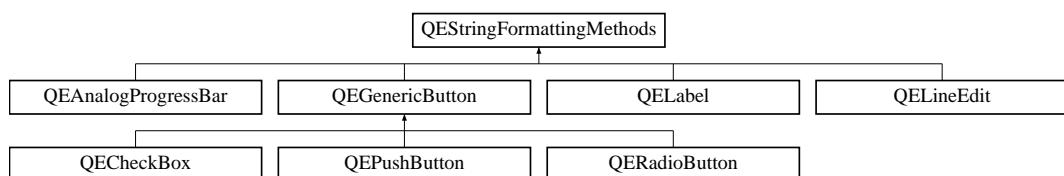
NOTATION_AUTOMATIC Automatic choice of standard or scientific notation.

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

- /tmp/epicsqt/trunk/framework/data/include/QEStringFormatting.h
- /tmp/epicsqt/trunk/framework/data/src/QEStringFormatting.cpp

9.127 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

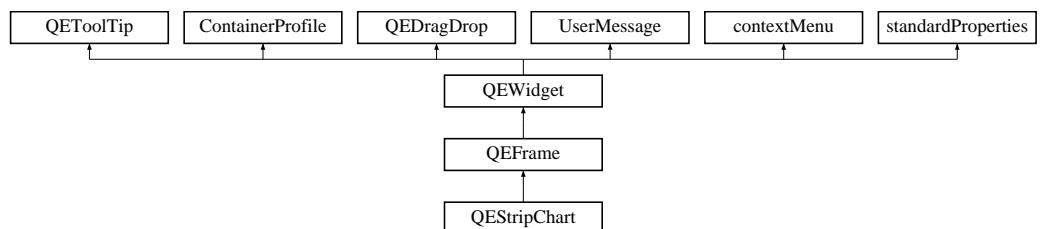
- QEStringFormatting **stringFormatting**

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

- /tmp/epicsqt/trunk/framework/widgets/include/QEStringFormattingMethods.h
- /tmp/epicsqt/trunk/framework/widgets/src/QEStringFormattingMethods.cpp

9.128 QEStripChart Class Reference

Inheritance diagram for QEStripChart:



Public Types

- enum **Constants** { **NUMBER_OF_PVS** = 12 }

Public Member Functions

- **QEStripChart** (QWidget *parent=0)
- QSize **sizeHint** () const
- QDateTime **getStartTime** ()
- QDateTime **getEndTime** ()
- void **setEndTime** (QDateTime endDateTimeIn)
- int **getDuration** ()
- void **setDuration** (int durationIn)
- double **getYMinimum** ()
- void **setYMinimum** (const double yMinimumIn)
- double **getYMaximum** ()
- void **setYMaximum** (const double yMaximumIn)
- void **setYRange** (const double yMinimumIn, const double yMaximumIn)

Protected Member Functions

- void **dragEnterEvent** (QDragEnterEvent *event)
- void **dropEvent** (QDropEvent *event)
- void **setDrop** (QVariant drop)
- void **paste** (QVariant s)
- void **setup** ()
- **qcaobject::QCaObject** * **createQcalItem** (unsigned int variableIndex)
- void **establishConnection** (unsigned int variableIndex)
- void **saveConfiguration** (**PersistanceManager** *pm)
- void **restoreConfiguration** (**PersistanceManager** *pm, **restorePhases** restorePhase)
- void **addToPredefinedList** (const QString &pvName)
- QStringList **getPredefinedPVNameList** ()
- QString **getPredefinedItem** (int i)
- void **setRecalcIsRequired** ()
- void **setReplotIsRequired** ()
- void **evaluateAllowDrop** ()

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`
- `QString variableSubstitutions`
- `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 [QEStripChartItem](#)

9.128.1 Member Function Documentation

**9.128.1.1 void QEStripChart::restoreConfiguration (`PersistanceManager *`,
`restorePhases`) [protected, virtual]**

Service a request to restore the QE widget's configuration. A QE widget recover any configuration details from the [PersistanceManager](#). For example, a [QEStripChart](#) may restore the variables being plotted. Many QE widgets do not have any persistant data requirements and do not implement this method. This is called twice with an incrementing restorePhase. Most widgets will miss the first call as they don't exist yet (they are created as part of the first phase)

Reimplemented from [QEWidget](#).

**9.128.1.2 void QEStripChart::saveConfiguration (`PersistanceManager *`)
[protected, virtual]**

Service a request to save the QE widget's current configuration. A widget may save any configuration details through the [PersistanceManager](#). For example, a [QEstripChart](#)

may save the variables being plotted. Many QE widgets do not have any persistant data requirements and do not implement this method.

Reimplemented from [QEWidget](#).

9.128.2 Property Documentation

9.128.2.1 QString QEStripChart::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.

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChart.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChart.cpp

9.129 QEStripChartAdjustPVDialog Class Reference

Public Member Functions

- **QEStripChartAdjustPVDialog** (QWidget *parent=0)
- void **setValueScaling** (const [ValueScaling](#) &valueScale)
- [ValueScaling](#) **getValueScaling** () const
- void **setSupport** (const double min, const double max, const QEDisplayRanges &loprHopr, const QEDisplayRanges &plotted, const QEDisplayRanges &buffered)

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartAdjustPVDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartAdjustPVDialog.cpp

9.130 QEStripChartContextMenu Class Reference

Signals

- void **contextMenuSelected** (const QEStripChartNames::ContextMenuOptions)

Public Member Functions

- [QEStripChartContextMenu](#) (bool inUse, QWidget *parent=0)
- void **setPredefinedNames** (const QStringList &pvList)

- void **setUseReceiveTime** (const bool useReceiveTime)
- void **setArchiveReadHow** (const QEArchiveInterface::How how)
- void **setLineDrawMode** (const QEStripChartNames::LineDrawModes mode)

9.130.1 Constructor & Destructor Documentation

9.130.1.1 QEStripChartContextMenu::QEStripChartContextMenu (bool *inUse*, QWidget * *parent* = 0) [explicit]

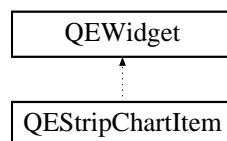
Construct strip chart item context menu. This menu item creates all required sub menu items. *inUse* set true for an inuse slot, i.e. already has a PV allocated. *inUse* set false for an empty slot.

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartContextMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartContextMenu.cpp

9.131 QEStripChartItem Class Reference

Inheritance diagram for QEStripChartItem:



Signals

- void **itemContextMenuRequested** (const unsigned int, const QPoint &)
- void **requestAction** (const QEActionRequests &)

Public Member Functions

- **QEStripChartItem** (QEStripChart *chart, unsigned int slot, QWidget *parent)
- bool **isInUse** ()
- bool **isCalculation** ()
- void **setPvName** (QString pvName, QString substitutions)
- QString **getPvName** ()
- bool **isScaled** ()
- bool **getUseReceiveTime** ()
- QEArchiveInterface::How **getArchiveReadHow** ()
- QEStripChartNames::LineDrawModes **getLineDrawMode** ()

- void **setColour** (const QColor &colour)
- QColor **getColour** ()
- QEDisplayRanges **getLoprHopr** (bool doScale)
- QEDisplayRanges **getDisplayedMinMax** (bool doScale)
- QEDisplayRanges **getBufferedMinMax** (bool doScale)
- QCaDataPointList **determinePlotPoints** ()
- void **readArchive** ()
- void **normalise** ()
- void **plotData** ()
- void **saveConfiguration** (PMElement &parentElement)
- void **restoreConfiguration** (PMElement &parentElement)

Public Attributes

- QCaVariableNamePropertyManager **pvNamePropertyManager**

Protected Member Functions

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

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartItem.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartItem.cpp

9.132 QEStripChartNames Class Reference

Public Types

- enum **ChartTimeModes** { **tmRealTime**, **tmPaused**, **tmHistorical** }
- enum **ChartYRanges** {
 manual, **operatingRange**, **plotted**, **buffered**,
 dynamic, **normalised** }
- enum **PlayModes** {
 play, **pause**, **forward**, **backward**,
 selectTimes }
- enum **StateModes** { **previous**, **next** }
- enum **VideoModes** { **normal**, **reverse** }
- enum **YScaleModes** { **linear**, **log** }
- enum **LineDrawModes** { **ldmHide**, **ldmRegular**, **ldmBold** }

- enum ContextMenuOptions {

SCCM_NONE = contextMenu::CM_SPECIFIC_WIDGETS_START_HERE, **SCCM_COPY_PV_NAMES**, **SCCM_PASTE_PV_NAMES**, **SCCM_READ_ARCHIVE**,

SCCM_SCALE_CHART_AUTO, **SCCM_SCALE_CHART_PLOTTED**, **SCCM_SCALE_CHART_BUFFERED**, **SCCM_SCALE_PV_RESET**,

SCCM_SCALE_PV_GENERAL, **SCCM_SCALE_PV_AUTO**, **SCCM_SCALE_PV_PLOTTED**, **SCCM_SCALE_PV_BUFFERED**,

SCCM_SCALE_PV_CENTRE, **SCCM_PLOT_RECTANGULAR**, **SCCM_PLOT_SMOOTH**, **SCCM_PLOT_SERVER_TIME**,

SCCM_PLOT_CLIENT_TIME, **SCCM_ARCH_LINEAR**, **SCCM_ARCH_PLOTBIN**, **SCCM_ARCH_RAW**,

SCCM_ARCH_SHEET, **SCCM_ARCH_AVERAGED**, **SCCM_LINE_HIDE**, **SCCM_LINE_REGULAR**,

SCCM_LINE_BOLD, **SCCM_LINE_COLOUR**, **SCCM_PV_EDIT_NAME**, **SCCM_ADD_TO_PREDEFINED**,

SCCM_PV_WRITE_TRACE, **SCCM_PV_STATS**, **SCCM_PV_CLEAR**, **SCCM_PV_ADD_NAME**,

SCCM_PV_PASTE_NAME, **SCCM_PREDEFINED_01**, **SCCM_PREDEFINED_02**, **SCCM_PREDEFINED_03**,

SCCM_PREDEFINED_04, **SCCM_PREDEFINED_05**, **SCCM_PREDEFINED_06**, **SCCM_PREDEFINED_07**,

SCCM_PREDEFINED_08, **SCCM_PREDEFINED_09**, **SCCM_PREDEFINED_10**
}

Static Public Attributes

- static const ContextMenuOptions **ContextMenuFirst** = **SCCM_READ_ARCHIVE**
- static const ContextMenuOptions **ContextMenuLast** = **SCCM_PREDEFINED_10**
- static const int **NumberPrefdefinedItems** = (**SCCM_PREDEFINED_10** - **SCCM_PREDEFINED_01** + 1)

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartNames.h

9.133 QEStripChartPushButtonSpecifications Struct Reference

Public Attributes

- int **gap**
- int **width**

- bool **isIcon**
- const QString **captionOrIcon**
- const QString **toolTip**
- const char * **member**

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartToolBar.cpp

9.134 QEStripChartRangeDialog Class Reference

Public Member Functions

- **QEStripChartRangeDialog** (QWidget *parent=0)
- void **setRange** (const double min, const double max)
- double **getMinimum** ()
- double **getMaximum** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartRangeDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartRangeDialog.cpp

9.135 QEStripChartState Class Reference

Public Member Functions

- void **saveConfiguration** ([PMElement](#) &parentElement)
- void **restoreConfiguration** ([PMElement](#) &parentElement)

Public Attributes

- bool **isNormalVideo**
- QEStripChartNames::ChartTimeModes **chartTimeMode**
- QEStripChartNames::YScaleModes **yScaleMode**
- QEStripChartNames::ChartYRanges **chartYScale**
- double **yMinimum**
- double **yMaximum**
- int **duration**
- Qt::TimeSpec **timeZoneSpec**
- QDateTime **endDateTime**

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartState.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartState.cpp

9.136 QEStripChartStateList Class Reference

Public Member Functions

- void **clear** ()
- void **push** (const QEStripChartState &state)
- bool **prev** (QEStripChartState &state)
- bool **next** (QEStripChartState &state)
- bool **prevAvailable** ()
- bool **nextAvailable** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartState.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartState.cpp

9.137 QEStripChartStatistics Class Reference

Public Member Functions

- **QEStripChartStatistics** (const QString &pvName, const QString &egu, const QCaDataPointList &dataList, QEStripChartItem *owner, QWidget *parent=0)

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartStatistics.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartStatistics.cpp

9.138 QEStripChartTimeDialog Class Reference

Public Member Functions

- **QEStripChartTimeDialog** (QWidget *parent=0)
- void **setMaximumDateTime** (QDateTime datetime)
- void **setStartTime** (QDateTime datetime)
- QDateTime **getStartTime** ()
- void **setEndDateTime** (QDateTime datetime)
- QDateTime **getEndDateTime** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartTimeDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartTimeDialog.cpp

9.139 QEStripChartToolBar Class Reference

This class holds all the StripChart tool bar widgets.

```
#include <QEStripChartToolBar.h>
```

Classes

- class [OwnWidgets](#)

Signals

- void **stateSelected** (const QEStripChartNames::StateModes mode)
- void **videoModeSelected** (const QEStripChartNames::VideoModes mode)
- void **yScaleModeSelected** (const QEStripChartNames::YScaleModes mode)
- void **yRangeSelected** (const QEStripChartNames::ChartYRanges scale)
- void **durationSelected** (const int seconds)
- void **timeZoneSelected** (const Qt::TimeSpec timeSpec)
- void **playModeSelected** (const QEStripChartNames::PlayModes mode)
- void **readArchiveSelected** ()

Public Member Functions

- **QEStripChartToolBar** (QWidget *parent=0)
- void **setYRangeStatus** (const QString &status)
- void **setTimeStatus** (const QString &timeStatus)
- void **setStateSelectionEnabled** (const QEStripChartNames::StateModes mode, const bool enabled)

Static Public Attributes

- static const int **designHeight** = 44

Protected Member Functions

- void **resizeEvent** (QResizeEvent *event)

9.139.1 Detailed Description

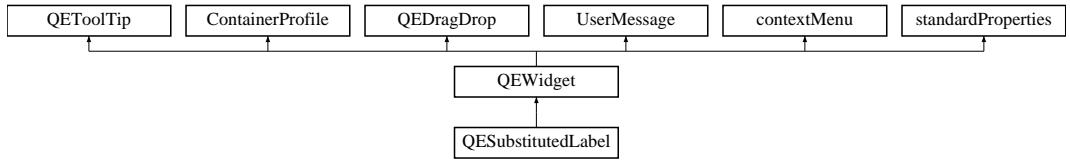
This class holds all the StripChart tool bar widgets.

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartToolBar.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartToolBar.cpp

9.140 QESubstitutedLabel Class Reference

Inheritance diagram for QESubstitutedLabel:



Public Member Functions

- **QESubstitutedLabel** (QWidget *parent=0)
- void **setLabelTextProperty** (QString labelTextIn)
- QString **getLabelTextProperty** ()
- void **setSubstitutionsProperty** (QString macroSubstitutionsIn)
- QString **getSubstitutionsProperty** ()
- QString **getLabelTextPropertyFormat** ()
- void **setLabelTextPropertyFormat** (QString labelTextIn)

Protected Attributes

- QString **labelText**

Properties

- QString **textSubstitutions**

9.140.1 Member Data Documentation

9.140.1.1 `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

9.140.2 Property Documentation

9.140.2.1 `QString QESubstitutedLabel::textSubstitutions` [read, write]

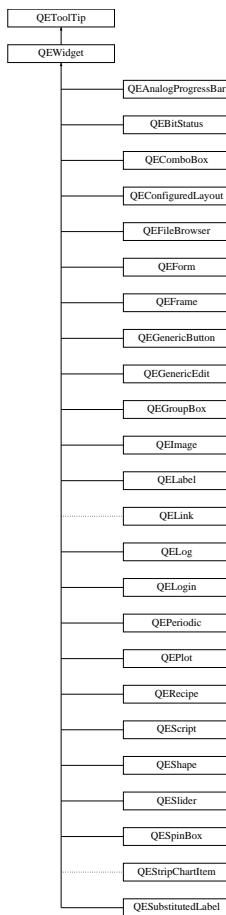
Text substitutions. These substitutions are applied to the 'labelText' property prior to copying it to the label text.

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

- /tmp/epicsqt/trunk/framework/widgets/QESubstitutedLabel/QESubstitutedLabel.h
- /tmp/epicsqt/trunk/framework/widgets/QESubstitutedLabel/QESubstitutedLabel.cpp

9.141 QEToolTip Class Reference

Inheritance diagram for QEToolTip:



Public Member Functions

- **QEToolTip** (QWidget *ownerIn)
- void **setNumberToolTipVariables** (const unsigned int number)
- void **updateToolTipVariable** (const QString &variable, const unsigned int variableIndex)
- void **updateToolTipAlarm** (const QString &alarm, const unsigned int variableIndex)
- void **updateToolTipConnection** (bool connection, const unsigned int variableIndex=0)

- void **updateToolTipCustom** (const QString &custom)
- void **setVariableAsToolTip** (const bool variableAsToolTip)
- bool **getVariableAsToolTip** () const

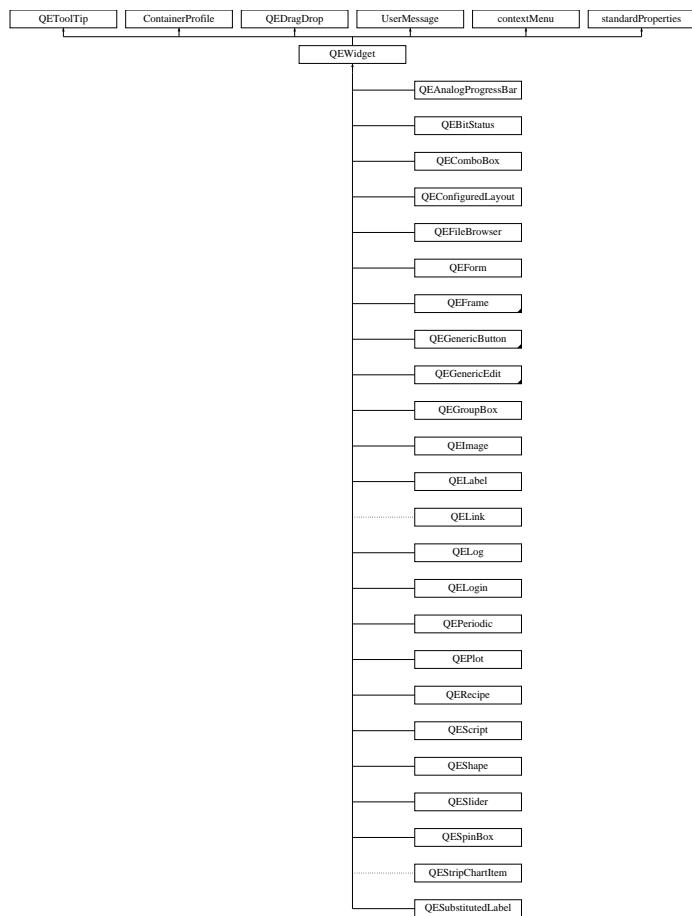
The documentation for this class was generated from the following files:

- /tmp/epicsqt/trunk/framework/widgets/include/QEToolTip.h
- /tmp/epicsqt/trunk/framework/widgets/src/QEToolTip.cpp

9.142 QEWidget Class Reference

```
#include <QEWidget.h>
```

Inheritance diagram for QEWidget:



Public Types

- enum `restorePhases` { **APPLICATION** = SaveRestoreSignal::RESTORE_APPLICATION, **FRAMEWORK** = SaveRestoreSignal::RESTORE_QEFramework }

Restore phases. When a widget's persistant data is restored, the restore occurs in two phases.

Public Member Functions

- `QEWidget (QWidget *ownerIn)`
Constructor.
- `virtual ~QEWidget ()`
Destructor.
- `void activate ()`
- `void deactivate ()`
- `unsigned int getMessageSourceld ()`
- `void setMessageSourceld (unsigned int messageSourceld)`
- `qcaobject::QCaObject * getQcalItem (unsigned int variableIndex)`
- `QColor getColor (QCaAlarmInfo &alarmInfo, const int saturation)`
- `void processAlarmInfo (QCaAlarmInfo &alarmInfo, const unsigned int variableIndex=0)`
- `void readNow ()`
- `virtual void writeNow ()`
- `virtual void setVariableNameAndSubstitutions (QString variableNameIn, QString variableNameSubstitutionsIn, unsigned int variableIndex)`
- `QFile * openQEFile (QString name, QFile::OpenModeFlag mode)`
- `QString defaultFileLocation ()`
- `QString getFrameworkVersion ()`
- `virtual void saveConfiguration (PersistanceManager *)`
- `virtual void restoreConfiguration (PersistanceManager *, restorePhases)`
- `virtual void scaleBy (const int, const int)`
- `QWidget * getQWidget ()`
- `virtual QMenu * getDefaultContextMenu ()`

Static Public Member Functions

- `static QFile * findQEFile (QString name, ContainerProfile *profile)`
- `static QFile * findQEFile (QString name)`
- `static void doAction (QWidget *searchPoint, QString widgetName, QString action, QStringList arguments, bool initialise, QAction *originator)`
- `static bool inDesigner ()`

Protected Member Functions

- void **setNumVariables** (unsigned int numVariablesIn)
- **qcaobject::QCaObject * createConnection** (unsigned int variableIndex)
- virtual **qcaobject::QCaObject * createQcalItem** (unsigned int variableIndex)
- virtual void **establishConnection** (unsigned int variableIndex)
- QString **persistentName** (QString prefix)
- virtual void **actionRequest** (QString, QStringList, bool, QAction *)
- void **deleteQcalItem** (unsigned int variableIndex, bool disconnect)

Protected Attributes

- bool **subscribe**

9.142.1 Detailed Description

This class is used as a base for all CA aware widgets, 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 [createQcalItem](#). 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 [VariableNameManager::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 its [establishConnection\(\)](#) function in one of two ways:

- 1) The variable name or variable name substitutions is changed by calling [setVariableName](#) 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 containing plugin definitions. After loading the plugin widgets,

code in the [QEForm](#) class calls the [activate\(\)](#) function in this class (QEWidget). 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 immediately 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 [createConnection\(\)](#), 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 from the [QEString](#) object to its [setLabelText](#) slot.

9.142.2 Member Function Documentation

9.142.2.1 void QEWidget::activate ()

Initiate updates. Called after all configuration is complete.

9.142.2.2 void QEWidget::deactivate ()

Terminates updates. This has been provided for third party (non QEGui) applications using the framework.

9.142.2.3 QString QEWidget::defaultFileLocation ()

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

9.142.2.4 void QEWidget::doAction (QWidget * *searchPoint*, QString *widgetName*, QString *action*, QStringList *arguments*, bool *initialise*, QAction * *originator*) [static]

Find a QE widget and request an action. The widget hierarchy under a supplied widget is searched for a QE widget with a given name. If found the QE widget will attempt to carry out the requested action which consists of an action string and an argument list. This method allows an application to initiate QE widget activity. The QEGui application uses this mechanism when providing custom menus defined in XML files. The method

returns true if the named widget was found. (The action was not necessarily performed, or even recognised by the widget)

9.142.2.5 QFile * QEWidget::findQEFile (QString *name*) [static]

Static method that looks for a file in a standard set of locations and assumes a current published profile. Returns a pointer to a QFile which is the caller's responsibility to delete, or NULL if the file was not found.

9.142.2.6 QFile * QEWidget::findQEFile (QString *name*, ContainerProfile * *profile*) [static]

Static method that looks for a file in a standard set of locations Returns a pointer to a QFile which is the caller's responsibility to delete, or NULL if the file was not found.

9.142.2.7 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

9.142.2.8 QString QEWidget::getFrameworkVersion ()

Returns the QE framework that built this instance of the widget. On windows, the QE-Framework DLL may be loaded twice with potentially different versions of it.

9.142.2.9 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.

9.142.2.10 qcaobject::QCaObject * QEWidget::getQcalItem (unsigned int *variableIndex*)

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

9.142.2.11 QWidget * QEWidget::getQWidget ()

Get the QWidget that the parent of this [QEWidget](#) instance is based on. For example, the parent of a [QEWidget](#) might be a [QELabel](#), which is based on QLabel which is based on QWidget.

```
9.142.2.12 QFile * QEWidget::openQEFile ( QString name, QFile::OpenModeFlag mode )
```

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

```
9.142.2.13 void QEWidget::processAlarmInfo ( QCaAlarmInfo & alarmInfo, const unsigned  
int variableIndex = 0 )
```

This convenience function updates the alarm tool tip, and alarm status style if the displayAlarmState property is set to true - assumes the widget uses standard properties. This function is perhaps most usefull for single-variable widgets.

```
9.142.2.14 void QEWidget::readNow ( )
```

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

```
9.142.2.15 virtual void QEWidget::restoreConfiguration ( PersistanceManager *,  
restorePhases ) [inline, virtual]
```

Service a request to restore the QE widget's configuration. A QE widget recover any configuration details from the [PersistanceManager](#). For example, a [QEStripChart](#) may restore the variables being plotted. Many QE widgets do not have any persistant data requirements and do not implement this method. This is called twice with an incrementing restorePhase. Most widgets will miss the first call as they don't exist yet (they are created as part of the first phase)

Reimplemented in [QEPvProperties](#), and [QEStripChart](#).

```
9.142.2.16 virtual void QEWidget::saveConfiguration ( PersistanceManager * )  
[inline, virtual]
```

Service a request to save the QE widget's current configuration. A widget may save any configuration details through the [PersistanceManager](#). For example, a [QEStripChart](#) may save the variables being plotted. Many QE widgets do not have any persistant data requirements and do not implement this method.

Reimplemented in [QEPvProperties](#), and [QEStripChart](#).

```
9.142.2.17 virtual void QEWidget::scaleBy ( const int , const int ) [inline,  
virtual]
```

Any [QEWidget](#) that requires additional scaling, i.e. above and beyond the standard scaling applied to size, minimum size, maximum size and font size, may override this function in order to perform any bespoke scaling need by the widget (for example see [QEShape](#)). The scaling is defined using a rational number specified by two integers (m, d). The first (m) parameter is the multiplier and the second (d) parameter is the divisor. For example, if m = 4 and d = 5, then an 80% scaling should be applied. And if m = 5 and d = 4, and a 125% scaling is required.

Reimplemented in [QEShape](#).

```
9.142.2.18 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.

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

Virtual function that may be implemented by users of [QEWidget](#) to update variable names and macro substitutions. A default is provided that is suitable in most cases.

Reimplemented in [QEBitStatus](#).

```
9.142.2.20 virtual void QEWidget::writeNow ( ) [inline, virtual]
```

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

Reimplemented in [QEGenericEdit](#).

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

- /tmp/epicsqt/trunk/framework/widgets/include/QEWidget.h
- /tmp/epicsqt/trunk/framework/widgets/src/QEWidget.cpp

9.143 QEWidgets Class Reference

Public Member Functions

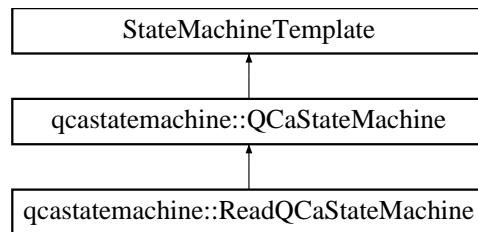
- **QEWidgets** (QObject *parent=0)
- virtual QList< QDesignerCustomWidgetInterface * > **customWidgets** () const

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

- /tmp/epicsqt/trunk/framework/widgets/include/QEDesignerPlugin.h
- /tmp/epicsqt/trunk/framework/widgets/src/QEDesignerPlugin.cpp

9.144 qcastatemachine::ReadQCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::ReadQCaStateMachine:



Public Member Functions

- **ReadQCaStateMachine** (void *parent)
- bool **process** (int requestedState)

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

- /tmp/epicsqt/trunk/framework/data/include/QCaStateMachine.h
- /tmp/epicsqt/trunk/framework/data/src/QCaStateMachine.cpp

9.145 recording Class Reference

Signals

- void **byteArrayChanged** (const QByteArray &value, unsigned long dataSize, QCaAlarmInfo &alarmInfo, QCaDateTime &timeStamp, const unsigned int &variableIndex)
- void **playingBack** (bool playing)

Public Member Functions

- **recording** (QWidget *parent=0)
- bool **isRecording** ()
- void **recordImage** (QByteArray image, unsigned long dataSize, QCaAlarmInfo &alarmInfo, QCaDateTime &time)
- void **nextFrameDue** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/recording.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/recording.cpp

9.146 imageDisplayProperties::rgbPixel Struct Reference

Public Attributes

- unsigned char **p** [4]

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/brightnessContrast.h

9.147 SaveRestoreSignal Class Reference

Public Types

- enum **saveRestoreOptions** { **SAVE**, **RESTORE_APPLICATION**, **RESTORE_QEFRAMEWORK** }

Signals

- void **saveRestore** (SaveRestoreSignal::saveRestoreOptions option)

Public Member Functions

- void **setOwner** ([PersistanceManager](#) *ownerIn)
- void **save** ()
- void **restore** ()

9.147.1 Member Function Documentation

9.147.1.1 void SaveRestoreSignal::restore ()

!! signal must be blocking

9.147.1.2 void SaveRestoreSignal::save ()

!! signal must be blocking

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

- /tmp/epicsqt/trunk/framework/widgets/include/persistanceManager.h
- /tmp/epicsqt/trunk/framework/widgets/src/persistanceManager.cpp

9.148 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 **setArea1Enabled** (bool enableAreaSelection)
- void **setArea2Enabled** (bool enableAreaSelection)
- void **setArea3Enabled** (bool enableAreaSelection)
- void **setArea4Enabled** (bool enableAreaSelection)
- void **setProfileEnabled** (bool enableProfileSelection)
- void **setTargetEnabled** (bool enableTargetSelection)
- void **setBeamEnabled** (bool enableBeamSelection)
- bool **getPanEnabled** ()
- bool **getVSliceEnabled** ()
- bool **getHSlicetEnabled** ()
- bool **getArea1Enabled** ()
- bool **getArea2Enabled** ()
- bool **getArea3Enabled** ()
- bool **getArea4Enabled** ()
- bool **getProfileEnabled** ()
- bool **getTargetEnabled** ()
- bool **getBeamEnabled** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QEImage/selectMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/selectMenu.cpp

9.149 signalSlotHandler Class Reference

Public Slots

- void **saveRestore** (SaveRestoreSignal::saveRestoreOptions option)

Public Member Functions

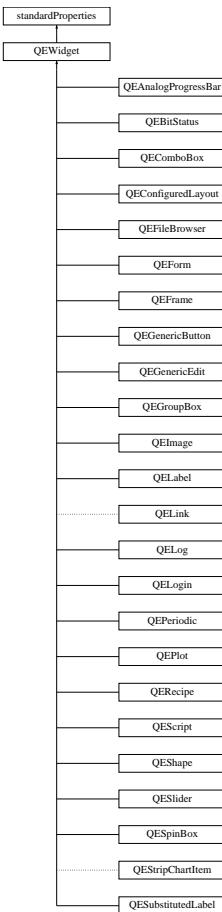
- void **setOwner** ([QEWidget](#) *ownerIn)

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

- /tmp/epicsqt/trunk/framework/widgets/include/QEWidget.h
- /tmp/epicsqt/trunk/framework/widgets/src/QEWidget.cpp

9.150 standardProperties Class Reference

Inheritance diagram for standardProperties:



Public Member Functions

- **standardProperties** (QWidget *ownerIn)
- **userLevelTypes::userLevels getUserLevelVisibility ()**
- **void setUserLevelVisibility (userLevelTypes::userLevels level)**
- **userLevelTypes::userLevels getUserLevelEnabled ()**
- **void setUserLevelEnabled (userLevelTypes::userLevels level)**
- **bool getApplicationEnabled () const**
- **void setApplicationEnabled (bool state)**
- **void setRunVisible (bool visibleIn)**
- **bool getRunVisible ()**
- **void setDisplayAlarmState (bool displayAlarmStateIn)**
- **bool getDisplayAlarmState ()**

Protected Member Functions

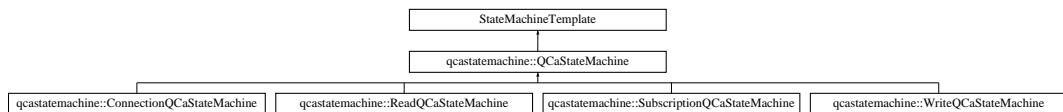
- void **checkVisibilityEnabledLevel** ([userLevelTypes::userLevels level](#))

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

- /tmp/epicsqt/trunk/framework/widgets/include/standardProperties.h
- /tmp/epicsqt/trunk/framework/widgets/src/standardProperties.cpp

9.151 StateMachineTemplate Class Reference

Inheritance diagram for StateMachineTemplate:



Public Member Functions

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

Public Attributes

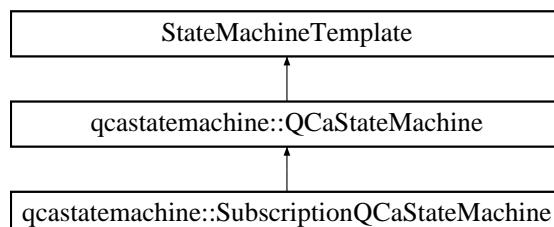
- int **currentState**
- int **requestState**

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

- /tmp/epicsqt/trunk/framework/data/include/QCaStateMachine.h

9.152 qcastatemachine::SubscriptionQCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::SubscriptionQCaStateMachine:



Public Member Functions

- **SubscriptionQCaStateMachine** (void *parent)
- bool **process** (int requestedState)

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

- /tmp/epicsqt/trunk/framework/data/include/QCaStateMachine.h
- /tmp/epicsqt/trunk/framework/data/src/QCaStateMachine.cpp

9.153 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:

- /tmp/epicsqt/trunk/framework/widgets/QEPlot/QEPlot.h

9.154 userInfoStruct Class Reference

Public Attributes

- bool **enable**
- double **value1**
- double **value2**
- QString **elementText**

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

- /tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodic.h

9.155 QEPeriodic::userInfoStructArray Struct Reference

Public Attributes

- `userInfoStruct array [NUM_ELEMENTS]`

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

- `/tmp/epicsqt/trunk/framework/widgets/QEPeriodic/QEPeriodic.h`

9.156 userLevelSignal Class Reference

Signals

- `void userChanged (userLevelTypes::userLevels level)`
Internal use only. Send when the user level has changed.

Public Member Functions

- `void setLevel (userLevelTypes::userLevels levelIn)`
- `userLevelTypes::userLevels getLevel ()`

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

- `/tmp/epicsqt/trunk/framework/widgets/include/ContainerProfile.h`
- `/tmp/epicsqt/trunk/framework/widgets/src/ContainerProfile.cpp`

9.157 userLevelSlot Class Reference

Public Slots

- `void userChanged (userLevelTypes::userLevels level)`

Public Member Functions

- `void setOwner (ContainerProfile *ownerIn)`

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

- `/tmp/epicsqt/trunk/framework/widgets/include/ContainerProfile.h`
- `/tmp/epicsqt/trunk/framework/widgets/src/ContainerProfile.cpp`

9.158 userLevelTypes Class Reference

Public Types

- enum `userLevels` { `USERLEVEL_USER`, `USERLEVEL_SCIENTIST`, `USERLEVEL_ENGINEER` }

9.158.1 Member Enumeration Documentation

9.158.1.1 enum userLevelTypes::userLevels

User levels set by widgets such as [QELogin](#) and used by many widgets to determine visibility, enabled state, and style.

Enumerator:

`USERLEVEL_USER` User level - least privilaged.

`USERLEVEL_SCIENTIST` User level - more privilaged than user, less than engineer.

`USERLEVEL_ENGINEER` User level - most privilaged.

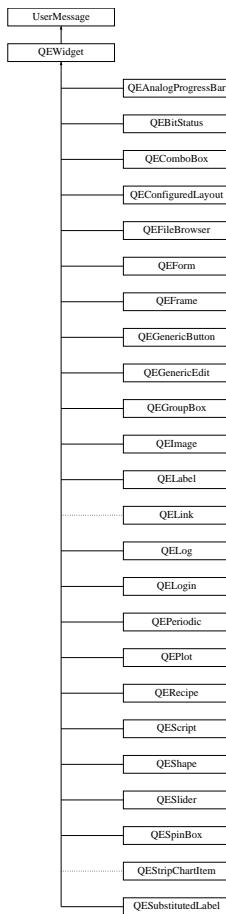
The documentation for this class was generated from the following file:

- /tmp/epicsqt/trunk/framework/widgets/include/ContainerProfile.h

9.159 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 **setSourceld** (unsigned int sourceld)

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 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 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_types(MESSAGE_TYPE_INFO))`
Send a message to the user.
- `void sendMessage (QString message, QString source, message_types type=message_types(MESSAGE_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 (typically logging widgets or application windows)

Friends

- class `UserMessageSlot`
- class `UserMessageSignal`

9.159.1 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 base a class on the [UserMessage](#) class and set up filtering to receive all messages. An application with log messages for separate 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:

- /tmp/epicsqt/trunk/framework/widgets/include/UserMessage.h
- /tmp/epicsqt/trunk/framework/widgets/src/UserMessage.cpp

9.160 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 receive 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.

9.160.1 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:

- /tmp/epicsqt/trunk/framework/widgets/include/UserMessage.h
- /tmp/epicsqt/trunk/framework/widgets/src/UserMessage.cpp

9.161 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.

9.161.1 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:

- /tmp/epicsqt/trunk/framework/widgets/include/UserMessage.h
- /tmp/epicsqt/trunk/framework/widgets/src/UserMessage.cpp

9.162 ValueScaling Class Reference

Public Member Functions

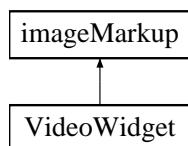
- void **reset** ()
- void **assign** (const [ValueScaling](#) &s)
- void **set** (const double dIn, const double mIn, const double cIn)
- void **get** (double &dOut, double &mOut, double &cOut) const
- void **map** (const double fromLower, const double fromUpper, const double toLower, const double toUpper)
- bool **isScaled** () const
- double **value** (const double x) const
- QEDisplayRanges **value** (const QEDisplayRanges &x) const
- void **saveConfiguration** ([PMElement](#) &parentElement) const
- void **restoreConfiguration** ([PMElement](#) &parentElement)

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartUtilities.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartUtilities.cpp

9.163 VideoWidget Class Reference

Inheritance diagram for VideoWidget:



Signals

- void **userSelection** (imageMarkup::markupIds mode, bool complete, bool clearing, QPoint point1, QPoint point2, unsigned int thickness)
- void **zoomInOut** (int zoomAmount)
- void **currentPixelInfo** (QPoint pos)
- void **pan** (QPoint pos)
- void **redraw** ()

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)
- QPoint **scaleImagePoint** (QPoint pnt)
- QRect **scaleImageRectangle** (QRect r)
- int **scaleImageOrdinate** (int ord)
- QImage **getImage** ()
- QSize **getImageSize** ()
- bool **hasCurrentImage** ()
- void **markupChange** ()

Protected Member Functions

- void **paintEvent** (QPaintEvent *)
- void **mousePressEvent** (QMouseEvent *event)
- void **mouseReleaseEvent** (QMouseEvent *event)
- void **mouseMoveEvent** (QMouseEvent *event)
- void **wheelEvent** (QWheelEvent *event)
- void **markupChange** (QVector< QRect > &changedAreas)
- void **resizeEvent** (QResizeEvent *event)
- void **markupSetCursor** (QCursor cursor)
- void **markupAction** (markupIds mode, bool complete, bool clearing, QPoint point1, QPoint point2, unsigned int thickness)

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

- /tmp/epicsqt/trunk/framework/widgets/QElImage/videowidget.h
- /tmp/epicsqt/trunk/framework/widgets/QElImage/videowidget.cpp

9.164 WidgetRef Class Reference

Public Member Functions

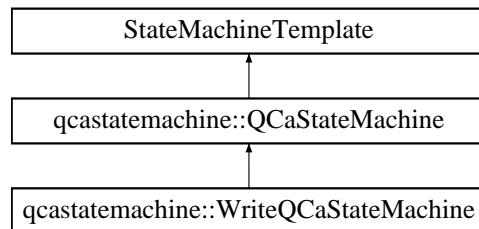
- **WidgetRef** (QEWidget *refIn)
- QEWidget * **getRef** ()

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

- /tmp/epicsqt/trunk/framework/widgets/include/ContainerProfile.h

9.165 qcastatemachine::WriteQCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::WriteQCaStateMachine:



Public Member Functions

- **WriteQCaStateMachine** (void *parent)
- bool **process** (int requestedState)

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

- /tmp/epicsqt/trunk/framework/data/include/QCaStateMachine.h
- /tmp/epicsqt/trunk/framework/data/src/QCaStateMachine.cpp

9.166 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:

- /tmp/epicsqt/trunk/framework/widgets/QEImage/zoomMenu.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/zoomMenu.cpp

Index

_CopyPaste, 29
_Field, 29
_Item, 30
_QDialogItem, 31
_QPushButtonGroup, 31
_QTableWidgetFileBrowser, 31
_QTableWidgetLog, 32
_QTableWidgetScript, 32

activate
 QEWidget, 303
addUnits
 QEAnalogProgressBar, 84
 QECheckBox, 100
 QELabel, 182
 QELineEdit, 190
 QENumericEdit, 205
 QEPushButton, 228
 QERadioButton, 246
alarmSeverityDisplayMode
 QEAnalogProgressBar, 84
alignment
 QECheckBox, 100
 QEPushButton, 228
 QERadioButton, 246
allowDrop
 QEAnalogProgressBar, 84
 QEBitStatus, 90
 QECheckBox, 100
 QEComboBox, 111
 QEFileBrowser, 120
 QEFrame, 127
 QEGenericEdit, 135
 QEGroupBox, 140
 QEImage, 160
 QELabel, 182
 QELog, 200
 QEPeriodic, 210
 QEPlot, 219
 QEPushButton, 228
 QERadioButton, 246
 QEShape, 268
 QESlider, 278
 QESpinBox, 282
 altReadbackVariable
 QEPushButton, 228
 animation1
 QEShape, 268
 animation2
 QEShape, 268
 animation3
 QEShape, 268
 animation4
 QEShape, 268
 animation5
 QEShape, 269
 animation6
 QEShape, 269
 animationOptions
 QEShape, 266
APPEND
 QEStringFormatting, 286
Append
 QEAnalogProgressBar, 82
 QECheckBox, 96
 QELabel, 180
 QELineEdit, 189
 QEPushButton, 225
 QERadioButton, 242
applicationLauncher, 32
areaColor
 QEImage, 160
arealInfo, 33
arguments
 QECheckBox, 100
 QEPushButton, 228
 QERadioButton, 246
arguments1
 QEImage, 160
arguments2
 QEImage, 161

arrayAction
QEAnalogProgressBar, 84
QECheckBox, 100
QELabel, 182
QELineEdit, 190
QEPushButton, 229
QERadioButton, 246

ArrayActions
QEAnalogProgressBar, 82
QECheckBox, 96
QELabel, 180
QELineEdit, 189
QEPushButton, 225
QERadioButton, 242

arrayActions
QEStringFormatting, 286

ASCII
QEStringFormatting, 286

Ascii
QEAnalogProgressBar, 82
QECheckBox, 96
QELabel, 180
QELineEdit, 189
QEPushButton, 225
QERadioButton, 242

autoBrightnessContrast
QEImage, 161

Automatic
QEAnalogProgressBar, 83
QECheckBox, 98
QELabel, 181
QELineEdit, 190
QEPushButton, 226
QERadioButton, 244

autoScale
QENumericEdit, 205

backgroundColour
QEAnalogIndicator, 77

Bar
QEAnalogIndicator, 77

Bayer
QEImage, 157

beamColor
QEImage, 161

beamXVariable
QEImage, 161

beamYVariable
QEImage, 161

bitDepthVariable

QEImage, 161
borderColour
QEAnalogIndicator, 77
Bottom_To_Top
QEAnalogIndicator, 77

briefInfoArea
QEImage, 161

centreAngle
QEAnalogIndicator, 77

clickCheckedText
QECheckBox, 100
QEPushButton, 229
QERadioButton, 246

clicked
QECheckBox, 99
QEPushButton, 227
QERadioButton, 245

clickText
QECheckBox, 101
QEPushButton, 229
QERadioButton, 247

clippingHighVariable
QEImage, 161

clippingLowVariable
QEImage, 161

clippingOnOffVariable
QEImage, 162

color1
QEShape, 269

color10
QEShape, 269

color2
QEShape, 269

color3
QEShape, 269

color4
QEShape, 269

color5
QEShape, 269

color6
QEShape, 269

color7
QEShape, 270

color8
QEShape, 270

color9
QEShape, 270

confirmAction
QECheckBox, 101

QEPushButton, 229
 QERadioButton, 247
 confirmText
 QECheckBox, 101
 QEPushButton, 229
 QERadioButton, 247
 confirmWrite
 QEGenericEdit, 135
 ContainerProfile, 35
 contextMenu, 36
 contextMenuObject, 38
 contrastReversal
 QEImage, 162
 creationOption
 QECheckBox, 101
 QEPushButton, 230
 QERadioButton, 247
 CreationOptionNames
 QECheckBox, 96
 QEPushButton, 225
 QERadioButton, 242
 customisationName
 QECheckBox, 101
 QEPushButton, 230
 QERadioButton, 247
 dbElementChanged
 QEPeriodic, 210
 dbValueChanged
 QEAnalogProgressBar, 83
 QEBitStatus, 90
 QECheckBox, 99
 QEComboBox, 111
 QEImage, 160
 QELabel, 182
 QELineEdit, 190
 QENumericEdit, 205
 QEPeriodic, 210
 QEPlot, 219
 QEPushButton, 227
 QERadioButton, 245
 QESlider, 277
 QESpinBox, 282
 dbValueChanged1
 QEShape, 267
 dbValueChanged2
 QEShape, 267
 dbValueChanged3
 QEShape, 267
 dbValueChanged4
 QEShape, 267
 dbValueChanged5
 QEShape, 268
 dbValueChanged6
 QEShape, 268
 deactivate
 QEWidget, 303
 Default
 QEAnalogProgressBar, 82
 QECheckBox, 97
 QELabel, 181
 QELineEdit, 189
 QEPushButton, 226
 QERadioButton, 243
 defaultFileLocation
 QEWidget, 303
 dimension1Variable
 QEImage, 162
 dimension2Variable
 QEImage, 162
 dimension3Variable
 QEImage, 162
 dimensionsVariable
 QEImage, 162
 displayAlarmState
 QEAnalogProgressBar, 84
 QEBitStatus, 90
 QECheckBox, 102
 QEComboBox, 111
 QEFileBrowser, 120
 QEFrame, 127
 QEGenericEdit, 135
 QEGroupBox, 140
 QEImage, 162
 QELabel, 183
 QELog, 200
 QEPeriodic, 210
 QEPlot, 219
 QEPushButton, 230
 QERadioButton, 248
 QEScript, 260
 QEShape, 270
 QESlider, 278
 QESpinBox, 282
 displayArea1Selection
 QEImage, 162
 displayArea2Selection
 QEImage, 163
 displayArea3Selection
 QEImage, 163

displayArea4Selection
QEImage, 163
displayBeamSelection
QEImage, 163
displayButtonBar
QEImage, 160
displayCursorPixelInfo
QEImage, 163
displayEllipse
QEImage, 163
displayHozSliceSelection
QEImage, 163
displayProfileSelection
QEImage, 163
displayTargetSelection
QEImage, 163
displayVertSliceSelection
QEImage, 164
doAction
QWidget, 303
DockBottom
QECheckBox, 97
QEPushButton, 225
QERadioButton, 243
DockBottomTabbed
QECheckBox, 97
QEPushButton, 225
QERadioButton, 243
DockFloating
QECheckBox, 97
QEPushButton, 225
QERadioButton, 243
DockLeft
QECheckBox, 97
QEPushButton, 225
QERadioButton, 243
DockLeftTabbed
QECheckBox, 97
QEPushButton, 225
QERadioButton, 243
DockRight
QECheckBox, 97
QEPushButton, 225
QERadioButton, 243
DockRightTabbed
QECheckBox, 97
QEPushButton, 225
QERadioButton, 243
DockTop
QECheckBox, 97
QEPushButton, 225
QERadioButton, 243
QECheckBox, 225
QERadioButton, 243
DockTopTabbed
QECheckBox, 97
QEPushButton, 225
QERadioButton, 243
drawMarkup
markupHLine, 51
markupVLine, 57
ellipseColor
QEImage, 164
ellipseX1Variable
QEImage, 164
ellipseX2Variable
QEImage, 164
ellipseY1Variable
QEImage, 164
ellipseY2Variable
QEImage, 164
enableArea1Selection
QEImage, 164
enableArea2Selection
QEImage, 164
enableArea3Selection
QEImage, 164
enableArea4Selection
QEImage, 165
enableBeamSelection
QEImage, 165
enableHozSliceSelection
QEImage, 165
enableProfileSelection
QEImage, 165
enableTargetSelection
QEImage, 165
enableVertSliceSelection
QEImage, 165
Engineer
QEAnalogProgressBar, 83
QEBitStatus, 90
QECheckBox, 98
QEComboBox, 110
QEFileBrowser, 120
QEFrame, 127
QEGenericEdit, 133
QEGroupBox, 139
QEImage, 159
QELabel, 182
QELog, 200

QEPeriodic, 210
 QEPlot, 218
 QEPushButton, 227
 QERadioButton, 244
 QEScript, 260
 QEShape, 267
 QESlider, 277
 QESpinBox, 282
 externalControls
 QEImage, 165

 FFBuffer, 39
 FFThread, 40
 findQEFile
 QEWidget, 304
 Fit
 QEImage, 158
 Fixed
 QEAnalogProgressBar, 83
 QECheckBox, 98
 QELabel, 181
 QELlineEdit, 190
 QEPpushButton, 226
 QERadioButton, 244
 flipRotateMenu, 40
 Floating
 QEAnalogProgressBar, 82
 QECheckBox, 97
 QELabel, 181
 QELlineEdit, 189
 QEPpushButton, 226
 QERadioButton, 243
 floating
 QCaDateTime, 69
 fontColour
 QEAnalogIndicator, 78
 foregroundColour
 QEAnalogIndicator, 78
 format
 QEAnalogProgressBar, 84
 QECheckBox, 102
 QELabel, 183
 QELlineEdit, 191
 QEPpushButton, 230
 QERadioButton, 248
 FORMAT_DEFAULT
 QEStringFormatting, 287
 FORMAT_FLOATING
 QEStringFormatting, 287
 FORMAT_INTEGER

QEStringFormatting, 287
 FORMAT_LOCAL_ENUMERATE
 QEStringFormatting, 287
 FORMAT_STRING
 QEStringFormatting, 287
 FORMAT_TIME
 QEStringFormatting, 287
 FORMAT_UNSIGNEDINTEGER
 QEStringFormatting, 287
 formatInteger
 QEIntegerFormatting, 176
 formatIntegerArray
 QEIntegerFormatting, 176
 formatOption
 QEImage, 166
 FormatOptions
 QEImage, 157
 Formats
 QEAnalogProgressBar, 82
 QECheckBox, 97
 QELabel, 180
 QELlineEdit, 189
 QEPpushButton, 225
 QERadioButton, 243
 formats
 QEStringFormatting, 287
 formatValue
 QEIntegerFormatting, 177
 formatVariable
 QEImage, 166
 fullScreenWindow, 40

 getColor
 QEWidget, 304
 getConfirmWrite
 QEGenericEdit, 134
 getElement
 PMElementList, 63
 getFrameworkVersion
 QEWidget, 304
 getLocalEnumeration
 QELocalEnumeration, 196
 getMessageSourceId
 QEWidget, 304
 getQcalItem
 QEWidget, 304
 getQWidget
 QEWidget, 304
 getSubscribe
 QEGenericEdit, 134

getWriteOnEnter
 QEGenericEdit, 134
getWriteOnFinish
 QEGenericEdit, 134
getWriteOnLoseFocus
 QEGenericEdit, 134
guiFile
 QECheckBox, 102
 QEPushButton, 230
 QERadioButton, 248

heightVariable
 QEImage, 166
histogram, 41
histogramScroll, 41
historicImage, 41
horizontalFlip
 QEImage, 166
hozSliceColor
 QEImage, 166

Icon
 QECheckBox, 98
 QEPushButton, 227
 QERadioButton, 244
imageContextMenu, 42
imageDisplayProperties, 43
imageDisplayProperties::rgbPixel, 307
imageInfo, 44
imageMarkup, 45
imageUpdateIndicator, 47
imageVariable
 QEImage, 166
INDEX
 QEStringFormatting, 287
Index
 QEAnalogProgressBar, 82
 QECheckBox, 96
 QELabel, 180
 QELineEdit, 189
 QEPushButton, 225
 QERadioButton, 242
initialHosScrollPos
 QEImage, 166
initialVertScrollPos
 QEImage, 160
int
 QEAnalogProgressBar, 84
 QEBitStatus, 90
 QECheckBox, 102
 QEComboBox, 111
 QEFileBrowser, 120
 QEFrame, 127
 QEGenericEdit, 135
 QEGroupBox, 140
 QEImage, 166
 QELabel, 183
 QELineEdit, 191
 QELog, 200
 QEPeriodic, 210
 QEPlot, 219
 QEPushButton, 230
 QERadioButton, 248
 QEScript, 260
 QEShape, 270
 QESlider, 278
 QESpinBox, 283
 Integer
 QEAnalogProgressBar, 82
 QECheckBox, 97
 QELabel, 181
 QELineEdit, 189
 QEPushButton, 226
 QERadioButton, 243
 isDefined
 QELocalEnumeration, 196
 labelText
 QECheckBox, 102
 QEPushButton, 231
 QERadioButton, 248
 QESubstitutedLabel, 298
 leadingZero
 QEAnalogProgressBar, 85
 QECheckBox, 103
 QELabel, 183
 QELineEdit, 191
 QEPushButton, 231
 QERadioButton, 249
 leadingZeros
 QNumericEdit, 205
 Left_To_Right
 QEAnalogIndicator, 77
lineProfileArrayVariable
 QEImage, 167
lineProfileThicknessVariable
 QEImage, 167
lineProfileX1Variable
 QEImage, 167
lineProfileX2Variable

QEImage, 167
lineProfileY1Variable
QEImage, 167
lineProfileY2Variable
QEImage, 167
LocalEnumeration
QEAnalogProgressBar, 82
QECheckBox, 97
QELabel, 181
QELlineEdit, 189
QEPushButton, 226
QERadioButton, 243
localEnumeration
QEAnalogProgressBar, 85
QECheckBox, 103
QEComboBox, 111
QELabel, 183
QELlineEdit, 191
QEPushButton, 231
QERadioButton, 249
logBrightness
QEImage, 167
loginWidget, 47
LogOutput
QECheckBox, 98
QEImage, 158
QEPushButton, 226
QERadioButton, 244
logScale
QEAnalogIndicator, 78
logScaleInterval
QEAnalogIndicator, 78
majorInterval
QEAnalogIndicator, 78
managePixmaps, 48
markupBeam, 48
markupDisplayMenu, 49
markupEllipse, 49
markupHLine, 50
drawMarkup, 51
markupItem, 51
markupLine, 53
markupRegion, 54
markupTarget, 55
markupText, 55
markupVLine, 56
drawMarkup, 57
maximum
QEAnalogIndicator, 78
QENumericEdit, 206
message_types, 57
Meter
QEAnalogIndicator, 77
minimum
QEAnalogIndicator, 78
QENumericEdit, 206
minorInterval
QEAnalogIndicator, 78
mode
QEAnalogIndicator, 78
Modes
QEAnalogIndicator, 77
Mono
QEImage, 157
mpegSource, 57
updateImage, 58
mpegSourceObject, 58
NewTab
QECheckBox, 97
QEPushButton, 225
QERadioButton, 243
NewWindow
QECheckBox, 97
QEPushButton, 225
QERadioButton, 243
None
QECheckBox, 98
QEImage, 158
QEPushButton, 226
QERadioButton, 244
NoRotation
QEImage, 158
notation
QEAnalogProgressBar, 85
QECheckBox, 103
QELabel, 184
QELlineEdit, 192
QEPushButton, 232
QERadioButton, 249
NOTATION_AUTOMATIC
QStringFormatting, 287
NOTATION_FIXED
QStringFormatting, 287
NOTATION_SCIENTIFIC
QStringFormatting, 287
Notations
QEAnalogProgressBar, 82
QECheckBox, 97

QELabel, 181
QELineEdit, 189
QEPushButton, 226
QERadioButton, 243
notations
 QEStringFormatting, 287

offset1
 QEShape, 270
offset2
 QEShape, 270
offset3
 QEShape, 271
offset4
 QEShape, 271
offset5
 QEShape, 271
offset6
 QEShape, 271
Open
 QECheckBox, 97
 QEPushButton, 225
 QERadioButton, 243
openQEFile
 QEWidget, 304
orientation
 QEAnalogIndicator, 78
Orientations
 QEAnalogIndicator, 77

password
 QECheckBox, 103
 QEPushButton, 232
 QERadioButton, 249
PeriodicDialog, 59
PeriodicElementSetupForm, 60
PeriodicSetupDialog, 60
PersistanceManager, 60
Picture
 QELabel, 181
 pixmap0
 QECheckBox, 104
 QELabel, 184
 QEPushButton, 232
 QERadioButton, 250
 pixmap1
 QECheckBox, 104
 QELabel, 184
 QEPushButton, 232
 QERadioButton, 250
 pixmap2
 QECheckBox, 104
 QELabel, 184
 QEPushButton, 232
 QERadioButton, 250
 pixmap3
 QECheckBox, 104
 QELabel, 185
 QEPushButton, 232
 QERadioButton, 250
 pixmap4
 QECheckBox, 104
 QELabel, 185
 QEPushButton, 232
 QERadioButton, 250
 pixmap5
 QECheckBox, 104
 QELabel, 185
 QEPushButton, 233
 QERadioButton, 250
 pixmap6
 QECheckBox, 104
 QELabel, 185
 QEPushButton, 233
 QERadioButton, 250
 pixmap7
 QECheckBox, 104
 QELabel, 185
 QEPushButton, 233
 QERadioButton, 250
playbackTimer, 61
PMContext, 61
PMElement, 62
PMElementList, 62
 getElement, 63
point1
 QEShape, 271
point10
 QEShape, 271
point2
 QEShape, 271
point3
 QEShape, 271
point4
 QEShape, 271
point5
 QEShape, 272
point6
 QEShape, 272
point7

QEShape, 272
 point8
 QEShape, 272
 point9
 QEShape, 272
 pointInfo, 63
 precision
 QEAnalogProgressBar, 86
 QECheckBox, 104
 QELabel, 185
 QELineEdit, 192
 QNumericUpDown, 206
 QEPushButton, 233
 QERadioButton, 250
 pressed
 QECheckBox, 99
 QEPushButton, 227
 QERadioButton, 245
 pressText
 QECheckBox, 105
 QEPushButton, 233
 QERadioButton, 251
 prioritySubstitutions
 QECheckBox, 105
 QEPushButton, 233
 QERadioButton, 251
 processAlarmInfo
 QEWidget, 305
 processManager, 63
 profileColor
 QEImage, 167
 profileHozArrayVariable
 QEImage, 167
 profileHozThicknessVariable
 QEImage, 168
 profileHozVariable
 QEImage, 168
 profilePlot, 64
 profileVertArrayVariable
 QEImage, 168
 profileVertThicknessVariable
 QEImage, 168
 profileVertVariable
 QEImage, 168
 program
 QECheckBox, 105
 QEPushButton, 233
 QERadioButton, 251
 program1
 QEImage, 168
 program2
 QEImage, 168
 programStartupOption
 QECheckBox, 105
 QEPushButton, 234
 QERadioButton, 251
 programStartupOption1
 QEImage, 168
 programStartupOption2
 QEImage, 169
 ProgramStartupOptionNames
 QECheckBox, 98
 QEImage, 157
 QEPushButton, 226
 QERadioButton, 244
 PublishedProfile, 64

 QBitStatus, 65
 QCaAlarmInfo, 67
 QCaConnectionInfo, 67
 QCaDataPoint, 68
 QCaDataPointList, 68
 QCaDateTime, 69
 floating, 69
 QCaEventFilter, 69
 QCaEventItem, 70
 QCaEventUpdate, 70
 QCaInstalledFiltersListItem, 70
 qcaobject::QCaObject, 71
 qcastatemachine::ConnectionQCaStateMachine,
 34
 qcastatemachine::QCaStateMachine, 73
 qcastatemachine::ReadQCaStateMachine,
 306
 qcastatemachine::SubscriptionQCaStateMachine,
 311
 qcastatemachine::WriteQCaStateMachine,
 321
 QCaVariableNamePropertyManager, 73
 QEAnalogIndicator, 74
 backgroundColour, 77
 Bar, 77
 borderColour, 77
 Bottom_To_Top, 77
 centreAngle, 77
 fontColour, 78
 foregroundColour, 78
 Left_To_Right, 77
 logScale, 78
 logScaleInterval, 78

majorInterval, 78
maximum, 78
Meter, 77
minimum, 78
minorInterval, 78
mode, 78
Modes, 77
orientation, 78
Orientations, 77
Right_To_Left, 77
Scale, 77
showScale, 79
showText, 79
spanAngle, 79
Top_To_Bottom, 77
value, 79
QEAnalogIndicator::Band, 34
QEAnalogIndicator::BandList, 34
QEAnalogProgressBar, 79
 addUnits, 84
 alarmSeverityDisplayMode, 84
 allowDrop, 84
 Append, 82
 arrayAction, 84
 ArrayActions, 82
 Ascii, 82
 Automatic, 83
 dbValueChanged, 83
 Default, 82
 displayAlarmState, 84
 Engineer, 83
 Fixed, 83
 Floating, 82
 format, 84
 Formats, 82
 Index, 82
 int, 84
 Integer, 82
 leadingZero, 85
 LocalEnumeration, 82
 localEnumeration, 85
 notation, 85
 Notations, 82
 precision, 86
 QEAnalogProgressBar, 83
 Scientific, 83
 Scientist, 83
 Time, 82
 trailingZeros, 86
 UnsignedInteger, 82
useDbDisplayLimits, 86
useDbPrecision, 86
User, 83
userLevelEnabled, 86
userLevelEngineerStyle, 86
UserLevels, 83
userLevelScientistStyle, 86
userLevelUserStyle, 87
userLevelVisibility, 87
variable, 87
variableAsToolTip, 87
variableSubstitutions, 87
visible, 87
QEBitStatus, 88
allowDrop, 90
dbValueChanged, 90
displayAlarmState, 90
Engineer, 90
int, 90
Scientist, 89
setVariableNameAndSubstitutions, 90
User, 89
userLevelEnabled, 90
userLevelEngineerStyle, 91
UserLevels, 89
userLevelScientistStyle, 91
userLevelUserStyle, 91
userLevelVisibility, 91
variable, 91
variableAsToolTip, 92
variableSubstitutions, 92
visible, 92
QEByteArray, 92
QECheckBox, 93
 addUnits, 100
 alignment, 100
 allowDrop, 100
 Append, 96
 arguments, 100
 arrayAction, 100
 ArrayActions, 96
 Ascii, 96
 Automatic, 98
 clickCheckedText, 100
 clicked, 99
 clickText, 101
 confirmAction, 101
 confirmText, 101
 creationOption, 101
 CreationOptionNames, 96

customisationName, 101
dbValueChanged, 99
Default, 97
displayAlarmState, 102
DockBottom, 97
DockBottomTabbed, 97
DockFloating, 97
DockLeft, 97
DockLeftTabbed, 97
DockRight, 97
DockRightTabbed, 97
DockTop, 97
DockTopTabbed, 97
Engineer, 98
Fixed, 98
Floating, 97
format, 102
Formats, 97
guiFile, 102
Icon, 98
Index, 96
int, 102
Integer, 97
labelText, 102
leadingZero, 103
LocalEnumeration, 97
localEnumeration, 103
LogOutput, 98
NewTab, 97
NewWindow, 97
None, 98
notation, 103
Notations, 97
Open, 97
password, 103
 pixmap0, 104
 pixmap1, 104
 pixmap2, 104
 pixmap3, 104
 pixmap4, 104
 pixmap5, 104
 pixmap6, 104
 pixmap7, 104
precision, 104
pressed, 99
pressText, 105
prioritySubstitutions, 105
program, 105
programStartupOption, 105
ProgramStartupOptionNames, 98
QECheckBox, 99
released, 99
releaseText, 105
requestAction, 99
Scientific, 98
Scientist, 98
State, 98
StdOutput, 98
subscribe, 105
Terminal, 98
Text, 98
TextAndIcon, 98
Time, 97
trailingZeros, 105
UnsignedInteger, 97
updateOption, 106
UpdateOptions, 98
useDbPrecision, 106
User, 98
userLevelEnabled, 106
userLevelEngineerStyle, 106
UserLevels, 98
userLevelScientistStyle, 106
userLevelUserStyle, 106
userLevelVisibility, 107
variable, 107
variableAsToolTip, 107
variableSubstitutions, 107
visible, 107
writeOnClick, 107
writeOnPress, 108
writeOnRelease, 108
QECheckBoxManager, 108
QEComboBox, 108
allowDrop, 111
dbValueChanged, 111
displayAlarmState, 111
Engineer, 110
int, 111
localEnumeration, 111
Scientist, 110
subscribe, 112
useDbEnumerations, 111
User, 110
userLevelEnabled, 112
userLevelEngineerStyle, 112
UserLevels, 110
userLevelScientistStyle, 112
userLevelUserStyle, 112
userLevelVisibility, 113

variable, 113
variableAsToolTip, 113
variableSubstitutions, 113
visible, 113
writeOnChange, 111
QEConfiguredLayout, 113
QEConfiguredLayoutManager, 115
QEDragDrop, 116
QEFileBrowser, 117
allowDrop, 120
displayAlarmState, 120
Engineer, 120
int, 120
Scientist, 120
selected, 120
User, 120
userLevelEnabled, 120
userLevelEngineerStyle, 121
UserLevels, 120
userLevelScientistStyle, 121
userLevelUserStyle, 121
userLevelVisibility, 121
variable, 121
variableAsToolTip, 122
variableSubstitutions, 122
visible, 122
QEFloating, 122
QEFloatingArray, 123
QEFloatingFormatting, 124
QEForm, 124
QEFrame, 126
allowDrop, 127
displayAlarmState, 127
Engineer, 127
int, 127
Scientist, 127
User, 127
userLevelEnabled, 127
userLevelEngineerStyle, 128
UserLevels, 127
userLevelScientistStyle, 128
userLevelUserStyle, 128
userLevelVisibility, 128
variableAsToolTip, 128
visible, 129
QEGenericButton, 129
QEGenericEdit, 131
allowDrop, 135
confirmWrite, 135
displayAlarmState, 135
Engineer, 133
getConfirmWrite, 134
getSubscribe, 134
getWriteOnEnter, 134
getWriteOnFinish, 134
getWriteOnLoseFocus, 134
int, 135
QEGenericEdit, 133
Scientist, 133
setConfirmWrite, 134
setSubscribe, 134
setWriteOnEnter, 134
setWriteOnFinish, 135
setWriteOnLoseFocus, 135
subscribe, 136
User, 133
userLevelEnabled, 136
userLevelEngineerStyle, 136
UserLevels, 133
userLevelScientistStyle, 136
userLevelUserStyle, 136
userLevelVisibility, 137
variable, 137
variableAsToolTip, 137
variableSubstitutions, 137
visible, 137
writeNow, 135
writeOnEnter, 137
writeOnFinish, 138
writeOnLoseFocus, 138
QEGroupBox, 138
allowDrop, 140
displayAlarmState, 140
Engineer, 139
int, 140
Scientist, 139
substitutedTitle, 140
textSubstitutions, 140
User, 139
userLevelEnabled, 140
userLevelEngineerStyle, 140
UserLevels, 139
userLevelScientistStyle, 141
userLevelUserStyle, 141
userLevelVisibility, 141
variableAsToolTip, 141
visible, 141
QEImage, 142
allowDrop, 160
areaColor, 160

arguments1, 160
arguments2, 161
autoBrightnessContrast, 161
Bayer, 157
beamColor, 161
beamXVariable, 161
beamYVariable, 161
bitDepthVariable, 161
briefInfoArea, 161
clippingHighVariable, 161
clippingLowVariable, 161
clippingOnOffVariable, 162
contrastReversal, 162
dbValueChanged, 160
dimension1Variable, 162
dimension2Variable, 162
dimension3Variable, 162
dimensionsVariable, 162
displayAlarmState, 162
displayArea1Selection, 162
displayArea2Selection, 163
displayArea3Selection, 163
displayArea4Selection, 163
displayBeamSelection, 163
displayButtonBar, 160
displayCursorPixelInfo, 163
displayEllipse, 163
displayHozSliceSelection, 163
displayProfileSelection, 163
displayTargetSelection, 163
displayVertSliceSelection, 164
ellipseColor, 164
ellipseX1Variable, 164
ellipseX2Variable, 164
ellipseY1Variable, 164
ellipseY2Variable, 164
enableArea1Selection, 164
enableArea2Selection, 164
enableArea3Selection, 164
enableArea4Selection, 165
enableBeamSelection, 165
enableHozSliceSelection, 165
enableProfileSelection, 165
enableTargetSelection, 165
enableVertSliceSelection, 165
Engineer, 159
externalControls, 165
Fit, 158
formatOption, 166
FormatOptions, 157
formatVariable, 166
heightVariable, 166
horizontalFlip, 166
hozSliceColor, 166
imageVariable, 166
initialHosScrollPos, 166
initialVertScrollPos, 160
int, 166
lineProfileArrayVariable, 167
lineProfileThicknessVariable, 167
lineProfileX1Variable, 167
lineProfileX2Variable, 167
lineProfileY1Variable, 167
lineProfileY2Variable, 167
logBrightness, 167
LogOutput, 158
Mono, 157
None, 158
NoRotation, 158
profileColor, 167
profileHozArrayVariable, 167
profileHozThicknessVariable, 168
profileHozVariable, 168
profileVertArrayVariable, 168
profileVertThicknessVariable, 168
profileVertVariable, 168
program1, 168
program2, 168
programStartupOption1, 168
programStartupOption2, 169
ProgramStartupOptionNames, 157
QEImage, 159, 160
regionOfInterest1HVariable, 169
regionOfInterest1WVariable, 169
regionOfInterest1XVariable, 169
regionOfInterest1YVariable, 169
regionOfInterest2HVariable, 169
regionOfInterest2WVariable, 169
regionOfInterest2XVariable, 169
regionOfInterest2YVariable, 170
regionOfInterest3HVariable, 170
regionOfInterest3WVariable, 170
regionOfInterest3XVariable, 170
regionOfInterest3YVariable, 170
regionOfInterest4HVariable, 170
regionOfInterest4WVariable, 170
regionOfInterest4XVariable, 170
regionOfInterest4YVariable, 170
RESIZE_OPTION_FIT, 158
RESIZE_OPTION_ZOOM, 158

resizeOption, 171
ResizeOptions, 158
resizeOptions, 158
rgb1, 157
rgb2, 157
rgb3, 157
Rotate180, 158
Rotate90Left, 158
Rotate90Right, 158
rotation, 171
ROTATION_0, 159
ROTATION_180, 159
ROTATION_90_LEFT, 159
ROTATION_90_RIGHT, 159
RotationOptions, 158
rotationOptions, 158
Scientist, 159
selectOptions, 159
showTime, 171
SO_AREA4, 159
SO_BEAM, 159
SO_HSLICE, 159
SO_NONE, 159
SO_PANNING, 159
SO_PROFILE, 159
SO_TARGET, 159
SO_VSLICE, 159
StdOutput, 158
targetColor, 171
targetTriggerVariable, 171
targetXVariable, 171
targetYVariable, 171
Terminal, 158
timeColor, 171
URL, 171
useFalseColour, 172
User, 159
userLevelEnabled, 172
userLevelEngineerStyle, 172
UserLevels, 159
userLevelScientistStyle, 172
userLevelUserStyle, 172
userLevelVisibility, 172
variableAsToolTip, 173
variableSubstitutions, 173
verticalFlip, 173
vertSliceColor, 173
visible, 173
widthVariable, 173
yuv422, 157
yuv444, 157
Zoom, 158
QEImageMarkupThickness, 174
QEImageOptionsDialog, 174
QEInteger, 174
QEIntegerArray, 175
QEIntegerFormatting, 176
formatInteger, 176
formatIntegerArray, 176
formatValue, 177
QELabel, 177
addUnits, 182
allowDrop, 182
Append, 180
arrayAction, 182
ArrayActions, 180
Ascii, 180
Automatic, 181
dbValueChanged, 182
Default, 181
displayAlarmState, 183
Engineer, 182
Fixed, 181
Floating, 181
format, 183
Formats, 180
Index, 180
int, 183
Integer, 181
leadingZero, 183
LocalEnumeration, 181
localEnumeration, 183
notation, 184
Notations, 181
Picture, 181
 pixmap0, 184
 pixmap1, 184
 pixmap2, 184
 pixmap3, 185
 pixmap4, 185
 pixmap5, 185
 pixmap6, 185
 pixmap7, 185
precision, 185
QELabel, 182
Scientific, 181
Scientist, 182
Text, 181
Time, 181
trailingZeros, 185

UnsignedInteger, 181
 UPDATE_PIXMAP, 181
 UPDATE_TEXT, 181
 updateOption, 185
 UpdateOptions, 181
 updateOptions, 181
 useDbPrecision, 186
 User, 182
 userLevelEnabled, 186
 userLevelEngineerStyle, 186
 UserLevels, 181
 userLevelScientistStyle, 186
 userLevelUserStyle, 186
 userLevelVisibility, 186
 variable, 187
 variableAsToolTip, 187
 variableSubstitutions, 187
 visible, 187
 QELineEdit, 187
 addUnits, 190
 Append, 189
 arrayAction, 190
 ArrayActions, 189
 Ascii, 189
 Automatic, 190
 dbValueChanged, 190
 Default, 189
 Fixed, 190
 Floating, 189
 format, 191
 Formats, 189
 Index, 189
 int, 191
 Integer, 189
 leadingZero, 191
 LocalEnumeration, 189
 localEnumeration, 191
 notation, 192
 Notations, 189
 precision, 192
 QELineEdit, 190
 Scientific, 190
 Time, 189
 trailingZeros, 192
 UnsignedInteger, 189
 useDbPrecision, 192
 QELineEditManager, 192
 QELink, 193
 QELocalEnumeration, 195
 getLocalEnumeration, 196
 isDefined, 196
 QELocalEnumeration, 195
 setLocalEnumeration, 196
 text.ToDouble, 196
 textToInt, 197
 textToValue, 197
 valueToText, 197
 QELog, 197
 allowDrop, 200
 displayAlarmState, 200
 Engineer, 200
 int, 200
 Scientist, 200
 User, 200
 userLevelEnabled, 200
 userLevelEngineerStyle, 200
 UserLevels, 199
 userLevelScientistStyle, 201
 userLevelUserStyle, 201
 userLevelVisibility, 201
 variableAsToolTip, 201
 visible, 201
 QELogin, 202
 QELoginDialog, 202
 QENumericEdit, 203
 addUnits, 205
 autoScale, 205
 dbValueChanged, 205
 leadingZeros, 205
 maximum, 206
 minimum, 206
 precision, 206
 QENumericEdit, 205
 QENumericEditManager, 206
 QEPeriodic, 207
 allowDrop, 210
 dbElementChanged, 210
 dbValueChanged, 210
 displayAlarmState, 210
 Engineer, 210
 int, 210
 readbackLabelVariable1, 211
 readbackLabelVariable2, 211
 Scientist, 210
 subscribe, 211
 User, 210
 userLevelEnabled, 211
 userLevelEngineerStyle, 211
 UserLevels, 210
 userLevelScientistStyle, 211

userLevelUserStyle, 212
userLevelVisibility, 212
variableAsToolTip, 212
variableSubstitutions, 212
visible, 212
writeButtonVariable1, 212
writeButtonVariable2, 213
QEPeriodic::elementInfoStruct, 39
QEPeriodic::userInfoStructArray, 313
QEPeriodicComponentData, 213
QEPeriodicTaskMenu, 213
QEPeriodicTaskMenuFactory, 214
QEpicsPV, 214
QEPlot, 215
 allowDrop, 219
 dbValueChanged, 219
 displayAlarmState, 219
 Engineer, 218
 int, 219
 Scientist, 218
 User, 218
 userLevelEnabled, 219
 userLevelEngineerStyle, 220
 UserLevels, 218
 userLevelScientistStyle, 220
 userLevelUserStyle, 220
 userLevelVisibility, 220
 variable1, 220
 variable2, 220
 variable3, 221
 variable4, 221
 variableAsToolTip, 221
 variableSubstitutions, 221
 visible, 221
QEPushButton, 221
 addUnits, 228
 alignment, 228
 allowDrop, 228
 altReadbackVariable, 228
 Append, 225
 arguments, 228
 arrayAction, 229
 ArrayActions, 225
 Ascii, 225
 Automatic, 226
 clickCheckedText, 229
 clicked, 227
 clickText, 229
 confirmAction, 229
 confirmText, 229
creationOption, 230
CreationOptionNames, 225
customisationName, 230
dbValueChanged, 227
Default, 226
displayAlarmState, 230
DockBottom, 225
DockBottomTabbed, 225
DockFloating, 225
DockLeft, 225
DockLeftTabbed, 225
DockRight, 225
DockRightTabbed, 225
DockTop, 225
DockTopTabbed, 225
Engineer, 227
Fixed, 226
Floating, 226
format, 230
Formats, 225
guiFile, 230
Icon, 227
Index, 225
int, 230
Integer, 226
labelText, 231
leadingZero, 231
LocalEnumeration, 226
localEnumeration, 231
LogOutput, 226
NewTab, 225
NewWindow, 225
None, 226
notation, 232
Notations, 226
Open, 225
password, 232
 pixmap0, 232
 pixmap1, 232
 pixmap2, 232
 pixmap3, 232
 pixmap4, 232
 pixmap5, 233
 pixmap6, 233
 pixmap7, 233
precision, 233
pressed, 227
pressText, 233
prioritySubstitutions, 233
program, 233

programStartupOption, 234
ProgramStartupOptionNames, 226
QEPushButton, 227
released, 228
releaseText, 234
requestAction, 228
Scientific, 226
Scientist, 227
State, 227
StdOutput, 226
subscribe, 234
Terminal, 226
Text, 227
TextAndIcon, 227
Time, 226
trailingZeros, 234
UnsignedInteger, 226
updateOption, 234
UpdateOptions, 226
useDbPrecision, 234
User, 227
userLevelEnabled, 234
userLevelEngineerStyle, 235
UserLevels, 227
userLevelScientistStyle, 235
userLevelUserStyle, 235
userLevelVisibility, 235
variable, 235
variableAsToolTip, 235
variableSubstitutions, 236
visible, 236
writeOnClick, 236
writeOnPress, 236
writeOnRelease, 236
QEPPVNameLists, 236
QEPPVProperties, 237
 restoreConfiguration, 238
 saveConfiguration, 238
 variable, 238
 variableSubstitutions, 238
QEPPVPropertiesManager, 239
QERadioButton, 239
 addUnits, 246
 alignment, 246
 allowDrop, 246
 Append, 242
 arguments, 246
 arrayAction, 246
 ArrayActions, 242
 Ascii, 242
Automatic, 244
clickCheckedText, 246
clicked, 245
clickText, 247
confirmAction, 247
confirmText, 247
creationOption, 247
CreationOptionNames, 242
customisationName, 247
dbValueChanged, 245
Default, 243
displayAlarmState, 248
DockBottom, 243
DockBottomTabbed, 243
DockFloating, 243
DockLeft, 243
DockLeftTabbed, 243
DockRight, 243
DockRightTabbed, 243
DockTop, 243
DockTopTabbed, 243
Engineer, 244
Fixed, 244
Floating, 243
format, 248
Formats, 243
guiFile, 248
Icon, 244
Index, 242
int, 248
Integer, 243
labelText, 248
leadingZero, 249
LocalEnumeration, 243
localEnumeration, 249
LogOutput, 244
NewTab, 243
NewWindow, 243
None, 244
notation, 249
Notations, 243
Open, 243
password, 249
 pixmap0, 250
 pixmap1, 250
 pixmap2, 250
 pixmap3, 250
 pixmap4, 250
 pixmap5, 250
 pixmap6, 250

pixmap7, 250
 precision, 250
 pressed, 245
 pressText, 251
 prioritySubstitutions, 251
 program, 251
 programStartupOption, 251
 ProgramStartupOptionNames, 244
 QERadioButton, 245
 released, 245
 releaseText, 251
 requestAction, 245
 Scientific, 244
 Scientist, 244
 State, 244
 StdOutput, 244
 subscribe, 251
 Terminal, 244
 Text, 244
 TextAndIcon, 244
 Time, 243
 trailingZeros, 251
 UnsignedInteger, 243
 updateOption, 252
 UpdateOptions, 244
 useDbPrecision, 252
 User, 244
 userLevelEnabled, 252
 userLevelEngineerStyle, 252
 UserLevels, 244
 userLevelScientistStyle, 252
 userLevelUserStyle, 252
 userLevelVisibility, 253
 variable, 253
 variableAsToolTip, 253
 variableSubstitutions, 253
 visible, 253
 writeOnClick, 253
 writeOnPress, 254
 writeOnRelease, 254
 QERecipe, 254
 QERecordFieldName, 256
 QERecordSpec, 257
 QERecordSpecList, 257
 QEScript, 257
 allowDrop, 260
 displayAlarmState, 260
 Engineer, 260
 int, 260
 Scientist, 260
 User, 260
 userLevelEnabled, 261
 userLevelEngineerStyle, 261
 UserLevels, 260
 userLevelScientistStyle, 261
 userLevelUserStyle, 261
 userLevelVisibility, 261
 variableAsToolTip, 262
 visible, 262
 QEShape, 262
 allowDrop, 268
 animation1, 268
 animation2, 268
 animation3, 268
 animation4, 268
 animation5, 269
 animation6, 269
 animationOptions, 266
 color1, 269
 color10, 269
 color2, 269
 color3, 269
 color4, 269
 color5, 269
 color6, 269
 color7, 270
 color8, 270
 color9, 270
 dbValueChanged1, 267
 dbValueChanged2, 267
 dbValueChanged3, 267
 dbValueChanged4, 267
 dbValueChanged5, 268
 dbValueChanged6, 268
 displayAlarmState, 270
 Engineer, 267
 int, 270
 offset1, 270
 offset2, 270
 offset3, 271
 offset4, 271
 offset5, 271
 offset6, 271
 point1, 271
 point10, 271
 point2, 271
 point3, 271
 point4, 271
 point5, 272
 point6, 272

point7, 272
point8, 272
point9, 272
QEShape, 267
scale2, 272
scale3, 272
scale4, 272
scale5, 272
scale6, 273
Scientist, 267
shapeOptions, 266
User, 267
userLevelEnabled, 273
userLevelEngineerStyle, 273
UserLevels, 266
userLevelScientistStyle, 273
userLevelUserStyle, 273
userLevelVisibility, 273
variable1, 274
variable2, 274
variable3, 274
variable4, 274
variable5, 274
variable6, 274
variableAsToolTip, 274
variableSubstitutions, 275
visible, 275
QESlider, 275
allowDrop, 278
dbValueChanged, 277
displayAlarmState, 278
Engineer, 277
int, 278
Scientist, 277
subscribe, 278
User, 277
userLevelEnabled, 278
userLevelEngineerStyle, 278
UserLevels, 277
userLevelScientistStyle, 279
userLevelUserStyle, 279
userLevelVisibility, 279
variable, 279
variableAsToolTip, 279
variableSubstitutions, 279
visible, 280
writeOnChange, 277
QESpinBox, 280
allowDrop, 282
dbValueChanged, 282
displayAlarmState, 282
Engineer, 282
int, 283
Scientist, 282
subscribe, 283
User, 282
userLevelEnabled, 283
userLevelEngineerStyle, 283
UserLevels, 282
userLevelScientistStyle, 283
userLevelUserStyle, 283
userLevelVisibility, 284
variable, 284
variableAsToolTip, 284
variableSubstitutions, 284
visible, 284
QEString, 285
QEStringFormatting, 285
APPEND, 286
arrayActions, 286
ASCII, 286
FORMAT_DEFAULT, 287
FORMAT_FLOATING, 287
FORMAT_INTEGER, 287
FORMAT_LOCAL_ENUMERATE, 287
FORMAT_STRING, 287
FORMAT_TIME, 287
FORMAT_UNSIGNEDINTEGER, 287
formats, 287
INDEX, 287
NOTATION_AUTOMATIC, 287
NOTATION_FIXED, 287
NOTATION_SCIENTIFIC, 287
notations, 287
QEStringFormattingMethods, 287
QEStripChart, 288
restoreConfiguration, 290
saveConfiguration, 290
variableSubstitutions, 291
QEStripChartAdjustPVDialog, 291
QEStripChartContextMenu, 291
QEStripChartContextMenu, 292
QEStripChartItem, 292
QEStripChartNames, 293
QEStripChartPushButtonSpecifications, 294
QEStripChartRangeDialog, 295
QEStripChartState, 295
QEStripChartStateList, 296
QEStripChartStatistics, 296
QEStripChartTimeDialog, 296

QEStripChartToolBar, 297
QEStripChartToolBar::OwnWidgets, 59
QESubstitutedLabel, 298
 labelText, 298
 textSubstitutions, 298
QEToolTip, 299
QEWidget, 300
 activate, 303
 deactivate, 303
 defaultFileLocation, 303
 doAction, 303
 findQEFile, 304
 getColor, 304
 getFrameworkVersion, 304
 getMessageSourceId, 304
 getQcalItem, 304
 getQWidget, 304
 openQEFile, 304
 processAlarmInfo, 305
 readNow, 305
 restoreConfiguration, 305
 saveConfiguration, 305
 scaleBy, 305
 setMessageSourceId, 306
 setVariableNameAndSubstitutions, 306
 writeNow, 306
QEWidgets, 306

readbackLabelVariable1
 QEPeriodic, 211
readbackLabelVariable2
 QEPeriodic, 211
readNow
 QWidget, 305
recording, 307
regionOfInterest1HVariable
 QEImage, 169
regionOfInterest1WVariable
 QEImage, 169
regionOfInterest1XVariable
 QEImage, 169
regionOfInterest1YVariable
 QEImage, 169
regionOfInterest2HVariable
 QEImage, 169
regionOfInterest2WVariable
 QEImage, 169
regionOfInterest2XVariable
 QEImage, 169
regionOfInterest2YVariable
 QEImage, 170
regionOfInterest3HVariable
 QEImage, 170
regionOfInterest3WVariable
 QEImage, 170
regionOfInterest3XVariable
 QEImage, 170
regionOfInterest3YVariable
 QEImage, 170
regionOfInterest4HVariable
 QEImage, 170
regionOfInterest4WVariable
 QEImage, 170
regionOfInterest4XVariable
 QEImage, 170
regionOfInterest4YVariable
 QEImage, 170
released
 QECheckBox, 99
 QEPushButton, 228
 QERadioButton, 245
releaseText
 QECheckBox, 105
 QEPushButton, 234
 QERadioButton, 251
requestAction
 QECheckBox, 99
 QEPushButton, 228
 QERadioButton, 245
RESIZE_OPTION_FIT
 QEImage, 158
RESIZE_OPTION_ZOOM
 QEImage, 158
resizeOption
 QEImage, 171
ResizeOptions
 QEImage, 158
resizeOptions
 QEImage, 158
restore
 SaveRestoreSignal, 308
restoreConfiguration
 QEPvProperties, 238
QEStripChart, 290
QEWidget, 305
rgb1
 QEImage, 157
rgb2
 QEImage, 157
rgb3

QEImage, 157
 Right_To_Left
 QEAnalogIndicator, 77
 Rotate180
 QEImage, 158
 Rotate90Left
 QEImage, 158
 Rotate90Right
 QEImage, 158
 rotation
 QEImage, 171
 ROTATION_0
 QEImage, 159
 ROTATION_180
 QEImage, 159
 ROTATION_90_LEFT
 QEImage, 159
 ROTATION_90_RIGHT
 QEImage, 159
 RotationOptions
 QEImage, 158
 rotationOptions
 QEImage, 158
 save
 SaveRestoreSignal, 308
 saveConfiguration
 QEPvProperties, 238
 QEStripChart, 290
 QWidget, 305
 SaveRestoreSignal, 308
 restore, 308
 save, 308
 Scale
 QEAnalogIndicator, 77
 scale2
 QEShape, 272
 scale3
 QEShape, 272
 scale4
 QEShape, 272
 scale5
 QEShape, 272
 scale6
 QEShape, 273
 scaleBy
 QWidget, 305
 Scientific
 QEAnalogProgressBar, 83
 QECheckBox, 98
 QELabel, 181
 QLineEdit, 190
 QEPushButton, 226
 QRadioButton, 244
 Scientist
 QEAnalogProgressBar, 83
 QEBitStatus, 89
 QECheckBox, 98
 QEComboBox, 110
 QEFileBrowser, 120
 QEFrame, 127
 QEGenericEdit, 133
 QEGroupBox, 139
 QEImage, 159
 QELabel, 182
 QELog, 200
 QEPeriodic, 210
 QEPlot, 218
 QEPushButton, 227
 QRadioButton, 244
 QEScript, 260
 QEShape, 267
 QESlider, 277
 QESpinBox, 282
 selected
 QEFileBrowser, 120
 selectMenu, 308
 selectOptions
 QEImage, 159
 setConfirmWrite
 QEGenericEdit, 134
 setLocalEnumeration
 QELocalEnumeration, 196
 setMessageSourceId
 QWidget, 306
 setSubscribe
 QEGenericEdit, 134
 setVariableNameAndSubstitutions
 QEBitStatus, 90
 QWidget, 306
 setWriteOnEnter
 QEGenericEdit, 134
 setWriteOnFinish
 QEGenericEdit, 135
 setWriteOnLoseFocus
 QEGenericEdit, 135
 shapeOptions
 QEShape, 266
 showScale
 QEAnalogIndicator, 79

showText
QEAnalogIndicator, 79

showTime
QEImage, 171

signalSlotHandler, 309

SO_AREA4
QEImage, 159

SO_BEAM
QEImage, 159

SO_HSLICE
QEImage, 159

SO_NONE
QEImage, 159

SO_PANNING
QEImage, 159

SO_PROFILE
QEImage, 159

SO_TARGET
QEImage, 159

SO_VSLICE
QEImage, 159

spanAngle
QEAnalogIndicator, 79

standardProperties, 310

State
QECheckBox, 98
QEPushButton, 227
QERadioButton, 244

StateMachineTemplate, 311

StdOutput
QECheckBox, 98
QEImage, 158
QEPushButton, 226
QERadioButton, 244

subscribe
QECheckBox, 105
QEComboBox, 112
QEGenericEdit, 136
QEPeriodic, 211
QEPushButton, 234
QERadioButton, 251
QESlider, 278
QESpinBox, 283

substitutedTitle
QEGroupBox, 140

targetColor
QEImage, 171

targetTriggerVariable
QEImage, 171

targetXVariable
QEImage, 171

targetYVariable
QEImage, 171

Terminal
QECheckBox, 98
QEImage, 158
QEPushButton, 226
QERadioButton, 244

Text
QECheckBox, 98
QELabel, 181
QEPushButton, 227
QERadioButton, 244

TextAndIcon
QECheckBox, 98
QEPushButton, 227
QERadioButton, 244

textSubstitutions
QEGroupBox, 140
QESubstitutedLabel, 298

textToDouble
QELocalEnumeration, 196

textToInt
QELocalEnumeration, 197

textToValue
QELocalEnumeration, 197

Time
QEAnalogProgressBar, 82
QECheckBox, 97
QELabel, 181
QELineEdit, 189
QEPushButton, 226
QERadioButton, 243

timeColor
QEImage, 171

Top_To_Bottom
QEAnalogIndicator, 77

trace, 312

trailingZeros
QEAnalogProgressBar, 86
QECheckBox, 105
QELabel, 185
QELineEdit, 192
QEPushButton, 234
QERadioButton, 251

UnsignedInteger
QEAnalogProgressBar, 82
QECheckBox, 97

QELabel, 181
QELineEdit, 189
QEPushButton, 226
QERadioButton, 243
UPDATE_PIXMAP
QELabel, 181
UPDATE_TEXT
QELabel, 181
updateImage
mpegSource, 58
updateOption
QECheckBox, 106
QELabel, 185
QEPushButton, 234
QERadioButton, 252
UpdateOptions
QECheckBox, 98
QELabel, 181
QEPushButton, 226
QERadioButton, 244
updateOptions
QELabel, 181
URL
QEImage, 171
useDbDisplayLimits
QEAnalogProgressBar, 86
useDbEnumerations
QEComboBox, 111
useDbPrecision
QEAnalogProgressBar, 86
QECheckBox, 106
QELabel, 186
QELineEdit, 192
QEPushButton, 234
QERadioButton, 252
useFalseColour
QEImage, 172
User
QEAnalogProgressBar, 83
QEBitStatus, 89
QECheckBox, 98
QEComboBox, 110
QEFileBrowser, 120
QEFrame, 127
QEGenericEdit, 133
QEGroupBox, 139
QEImage, 159
QELabel, 182
QELog, 200
QEPeriodic, 210
QEPlot, 218
QEPushButton, 227
QERadioButton, 244
QEScript, 260
QEShape, 267
QESlider, 277
QESpinBox, 282
userInfoStruct, 312
USERLEVEL_ENGINEER
userLevelTypes, 314
USERLEVEL_SCIENTIST
userLevelTypes, 314
USERLEVEL_USER
userLevelTypes, 314
userLevelEnabled
QEAnalogProgressBar, 86
QEBitStatus, 90
QECheckBox, 106
QEComboBox, 112
QEFileBrowser, 120
QEFrame, 127
QEGenericEdit, 136
QEGroupBox, 140
QEImage, 172
QELabel, 186
QELog, 200
QEPeriodic, 211
QEPlot, 219
QEPushButton, 234
QERadioButton, 252
QEScript, 261
QEShape, 273
QESlider, 278
QESpinBox, 283
userLevelEngineerStyle
QEAnalogProgressBar, 86
QEBitStatus, 91
QECheckBox, 106
QEComboBox, 112
QEFileBrowser, 121
QEFrame, 128
QEGenericEdit, 136
QEGroupBox, 140
QEImage, 172
QELabel, 186
QELog, 200
QEPeriodic, 211
QEPlot, 220
QEPushButton, 235
QERadioButton, 252

QEScript, 261
QEShape, 273
QESlider, 278
QESpinBox, 283
UserLevels
QEAnalogProgressBar, 83
QEBitStatus, 89
QECheckBox, 98
QEComboBox, 110
QEFileBrowser, 120
QEFrame, 127
QEGenericEdit, 133
QEGroupBox, 139
QEImage, 159
QELabel, 181
QELog, 199
QEPeriodic, 210
QEPlot, 218
QEPushButton, 227
QERadioButton, 244
QEScript, 260
QEShape, 266
QESlider, 277
QESpinBox, 282
userLevels
userLevelTypes, 314
userLevelScientistStyle
QEAnalogProgressBar, 86
QEBitStatus, 91
QECheckBox, 106
QEComboBox, 112
QEFileBrowser, 121
QEFrame, 128
QEGenericEdit, 136
QEGroupBox, 141
QEImage, 172
QELabel, 186
QELog, 201
QEPeriodic, 212
QEPlot, 220
QEPushButton, 235
QERadioButton, 252
QEScript, 261
QEShape, 273
QESlider, 279
QESpinBox, 283
userLevelSignal, 313
userLevelSlot, 313
userLevelTypes, 314
USERLEVEL_ENGINEER, 314
USERLEVEL_SCIENTIST, 314
USERLEVEL_USER, 314
userLevels, 314
userLevelUserStyle
QEAnalogProgressBar, 87
QEBitStatus, 91
QECheckBox, 106
QEComboBox, 112
QEFileBrowser, 121
QEFrame, 128
QEGenericEdit, 136
QEGroupBox, 141
QEImage, 172
QELabel, 186
QELog, 201
QEPeriodic, 212
QEPlot, 220
QEPushButton, 235
QERadioButton, 252
QEScript, 261
QEShape, 273
QESlider, 279
QESpinBox, 283
userLevelVisibility
QEAnalogProgressBar, 87
QEBitStatus, 91
QECheckBox, 107
QEComboBox, 113
QEFileBrowser, 121
QEFrame, 128
QEGenericEdit, 137
QEGroupBox, 141
QEImage, 172
QELabel, 186
QELog, 201
QEPeriodic, 212
QEPlot, 220
QEPushButton, 235
QERadioButton, 253
QEScript, 261
QEShape, 273
QESlider, 279
QESpinBox, 284
UserMessage, 314
UserMessageSignal, 317
UserMessageSlot, 318
value
QEAnalogIndicator, 79
ValueScaling, 319

valueToText
 QELocalEnumeration, 197
 variable
 QEAnalogProgressBar, 87
 QEBitStatus, 91
 QECheckBox, 107
 QEComboBox, 113
 QEFileBrowser, 121
 QEGenericEdit, 137
 QELabel, 187
 QEPushButton, 235
 QE PvProperties, 238
 QERadioButton, 253
 QESlider, 279
 QESpinBox, 284
 variable1
 QEPlot, 220
 QEShape, 274
 variable2
 QEPlot, 220
 QEShape, 274
 variable3
 QEPlot, 221
 QEShape, 274
 variable4
 QEPlot, 221
 QEShape, 274
 variable5
 QEShape, 274
 variable6
 QEShape, 274
 variableAsToolTip
 QEAnalogProgressBar, 87
 QEBitStatus, 92
 QECheckBox, 107
 QEComboBox, 113
 QEFileBrowser, 122
 QEFrame, 128
 QEGenericEdit, 137
 QEGroupBox, 141
 QEImage, 173
 QELabel, 187
 QELog, 201
 QEPeriodic, 212
 QEPlot, 221
 QEPushButton, 235
 QERadioButton, 253
 QE Script, 262
 QEShape, 274
 QESlider, 279
 QESpinBox, 284
 WidgetRef, 320
 widthVariable
 QEImage, 173
 writeButtonVariable1
 QEPeriodic, 212

writeButtonVariable2
QEPeriodic, 213
writeNow
QEGenericEdit, 135
QEWidget, 306
writeOnChange
QEComboBox, 111
QESlider, 277
writeOnClick
QECheckBox, 107
QEPushButton, 236
QERadioButton, 253
writeOnEnter
QEGenericEdit, 137
writeOnFinish
QEGenericEdit, 138
writeOnLoseFocus
QEGenericEdit, 138
writeOnPress
QECheckBox, 108
QEPushButton, 236
QERadioButton, 254
writeOnRelease
QECheckBox, 108
QEPushButton, 236
QERadioButton, 254
yuv422
QEImage, 157
yuv444
QEImage, 157
Zoom
QEImage, 158
zoomMenu, 321