

# EPICS QT Framework

## 2.6.0

Generated by Doxygen 1.7.4

Mon Jul 1 2013 10:02:07



# Contents

<b>1</b>	<b>QE framework - EPICS aware Qt Widgets and data access classes</b>	<b>1</b>
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
<b>2</b>	<b>GNU General Public License</b>	<b>5</b>
<b>3</b>	<b>ASgui screen shots</b>	<b>7</b>
<b>4</b>	<b>other applications using epicsqt widgets</b>	<b>13</b>
<b>5</b>	<b>Qt Designer</b>	<b>15</b>
<b>6</b>	<b>Qt Creator</b>	<b>17</b>
<b>7</b>	<b>Class Index</b>	<b>19</b>
7.1	Class Hierarchy . . . . .	19
<b>8</b>	<b>Class Index</b>	<b>23</b>
8.1	Class List . . . . .	23
<b>9</b>	<b>Class Documentation</b>	<b>27</b>
9.1	_Field Class Reference . . . . .	27

9.2	<a href="#">_Item Class Reference</a>	28
9.3	<a href="#">_QDialogItem Class Reference</a>	28
9.4	<a href="#">_QPushButtonGroup Class Reference</a>	28
9.5	<a href="#">_QTableWidgetFileBrowser Class Reference</a>	29
9.6	<a href="#">_QTableWidgetLog Class Reference</a>	29
9.7	<a href="#">_QTableWidgetScript Class Reference</a>	29
9.8	<a href="#">QEAnalogIndicator::Band Struct Reference</a>	30
9.9	<a href="#">QEAnalogIndicator::BandList Class Reference</a>	30
9.10	<a href="#">ChartState Class Reference</a>	30
9.11	<a href="#">qcastatemachine::ConnectionQCaStateMachine Class Reference</a>	31
9.12	<a href="#">ContainerProfile Class Reference</a>	31
9.13	<a href="#">contextMenu Class Reference</a>	33
9.14	<a href="#">contextMenuObject Class Reference</a>	35
9.15	<a href="#">QEPeriodic::elementInfoStruct Struct Reference</a>	35
9.16	<a href="#">flipRotateMenu Class Reference</a>	36
9.17	<a href="#">imageContextMenu Class Reference</a>	36
9.18	<a href="#">imageInfo Class Reference</a>	37
9.19	<a href="#">imageMarkup Class Reference</a>	38
9.20	<a href="#">loginWidget Class Reference</a>	39
9.21	<a href="#">managePixmap Class Reference</a>	40
9.22	<a href="#">markupBeam Class Reference</a>	40
9.23	<a href="#">markupHLine Class Reference</a>	41
9.23.1	<a href="#">Member Function Documentation</a>	42
9.23.1.1	<a href="#">drawMarkup</a>	42
9.24	<a href="#">markupItem Class Reference</a>	42
9.25	<a href="#">markupLine Class Reference</a>	44
9.26	<a href="#">markupRegion Class Reference</a>	45
9.27	<a href="#">markupTarget Class Reference</a>	45
9.28	<a href="#">markupText Class Reference</a>	46
9.29	<a href="#">markupVLine Class Reference</a>	47
9.29.1	<a href="#">Member Function Documentation</a>	48
9.29.1.1	<a href="#">drawMarkup</a>	48
9.30	<a href="#">message_types Class Reference</a>	48
9.31	<a href="#">QEStripChartToolBar::OwnWidgets Class Reference</a>	48

9.32 QEPvProperties::OwnWidgets Class Reference . . . . .	49
9.33 PeriodicDialog Class Reference . . . . .	49
9.34 PeriodicElementSetupForm Class Reference . . . . .	50
9.35 PeriodicSetupDialog Class Reference . . . . .	50
9.36 PersistenceManager Class Reference . . . . .	50
9.37 PMContext Class Reference . . . . .	51
9.38 PMElement Class Reference . . . . .	51
9.39 PMElementList Class Reference . . . . .	52
9.39.1 Member Function Documentation . . . . .	52
9.39.1.1 getElement . . . . .	52
9.40 QEStripChart::PrivateData Class Reference . . . . .	52
9.41 QEStripChartItem::PrivateData Class Reference . . . . .	53
9.42 profilePlot Class Reference . . . . .	53
9.43 PublishedProfile Class Reference . . . . .	54
9.44 QPushButtonSpecifications Struct Reference . . . . .	54
9.45 QBitStatus Class Reference . . . . .	55
9.46 QCaAlarmInfo Class Reference . . . . .	56
9.47 QCaConnectionInfo Class Reference . . . . .	57
9.48 QCaDataPoint Class Reference . . . . .	57
9.49 QCaDataPointList Class Reference . . . . .	58
9.50 QCaDateTime Class Reference . . . . .	58
9.50.1 Member Function Documentation . . . . .	58
9.50.1.1 floating . . . . .	58
9.51 QCaEventFilter Class Reference . . . . .	58
9.52 QCaEventItem Class Reference . . . . .	59
9.53 QCaEventUpdate Class Reference . . . . .	59
9.54 QCaInstalledFiltersListItem Class Reference . . . . .	60
9.55 qcaobject::QCaObject Class Reference . . . . .	60
9.56 qcastatemachine::QCaStateMachine Class Reference . . . . .	62
9.57 QCaVariableNamePropertyManager Class Reference . . . . .	62
9.58 QEAnalogIndicator Class Reference . . . . .	63
9.58.1 Detailed Description . . . . .	66
9.58.2 Member Enumeration Documentation . . . . .	66
9.58.2.1 Modes . . . . .	66

9.58.2.2	Orientations	66
9.58.3	Property Documentation	66
9.58.3.1	backgroundColour	66
9.58.3.2	borderColour	66
9.58.3.3	centreAngle	66
9.58.3.4	fontColour	67
9.58.3.5	foregroundColour	67
9.58.3.6	logScale	67
9.58.3.7	logScaleInterval	67
9.58.3.8	majorInterval	67
9.58.3.9	maximum	67
9.58.3.10	minimum	67
9.58.3.11	minorInterval	67
9.58.3.12	mode	67
9.58.3.13	orientation	67
9.58.3.14	showScale	68
9.58.3.15	showText	68
9.58.3.16	spanAngle	68
9.58.3.17	value	68
9.59	QEAAnalogProgressBar Class Reference	68
9.59.1	Member Enumeration Documentation	71
9.59.1.1	ArrayActions	71
9.59.1.2	Formats	71
9.59.1.3	Notations	71
9.59.1.4	UserLevels	72
9.59.2	Constructor & Destructor Documentation	72
9.59.2.1	QEAAnalogProgressBar	72
9.59.2.2	QEAAnalogProgressBar	72
9.59.3	Member Function Documentation	72
9.59.3.1	dbValueChanged	72
9.59.4	Property Documentation	72
9.59.4.1	addUnits	72
9.59.4.2	alarmSeverityDisplayMode	72
9.59.4.3	allowDrop	73

9.59.4.4	arrayAction	73
9.59.4.5	displayAlarmState	73
9.59.4.6	format	73
9.59.4.7	int	73
9.59.4.8	leadingZero	74
9.59.4.9	localEnumeration	74
9.59.4.10	notation	74
9.59.4.11	precision	74
9.59.4.12	trailingZeros	75
9.59.4.13	useDbDisplayLimits	75
9.59.4.14	useDbPrecision	75
9.59.4.15	userLevelEnabled	75
9.59.4.16	userLevelEngineerStyle	75
9.59.4.17	userLevelScientistStyle	75
9.59.4.18	userLevelUserStyle	75
9.59.4.19	userLevelVisibility	76
9.59.4.20	variable	76
9.59.4.21	variableAsToolTip	76
9.59.4.22	variableSubstitutions	76
9.59.4.23	visible	76
9.60	QEBitStatus Class Reference	77
9.60.1	Member Enumeration Documentation	78
9.60.1.1	UserLevels	78
9.60.2	Member Function Documentation	78
9.60.2.1	dbValueChanged	78
9.60.2.2	setVariableNameAndSubstitutions	79
9.60.3	Property Documentation	79
9.60.3.1	allowDrop	79
9.60.3.2	displayAlarmState	79
9.60.3.3	int	79
9.60.3.4	userLevelEnabled	79
9.60.3.5	userLevelEngineerStyle	79
9.60.3.6	userLevelScientistStyle	80
9.60.3.7	userLevelUserStyle	80

9.60.3.8	<a href="#">userLevelVisibility</a>	80
9.60.3.9	<a href="#">variable</a>	80
9.60.3.10	<a href="#">variableAsToolTip</a>	80
9.60.3.11	<a href="#">variableSubstitutions</a>	81
9.60.3.12	<a href="#">visible</a>	81
9.61	<a href="#">QEByteArray Class Reference</a>	81
9.62	<a href="#">QEChartStateLists Class Reference</a>	82
9.63	<a href="#">QECheckBox Class Reference</a>	82
9.63.1	<a href="#">Member Enumeration Documentation</a>	85
9.63.1.1	<a href="#">ArrayActions</a>	85
9.63.1.2	<a href="#">CreationOptionNames</a>	85
9.63.1.3	<a href="#">Formats</a>	85
9.63.1.4	<a href="#">Notations</a>	86
9.63.1.5	<a href="#">UpdateOptions</a>	86
9.63.1.6	<a href="#">UserLevels</a>	86
9.63.2	<a href="#">Constructor &amp; Destructor Documentation</a>	87
9.63.2.1	<a href="#">QECheckBox</a>	87
9.63.2.2	<a href="#">QECheckBox</a>	87
9.63.3	<a href="#">Member Function Documentation</a>	87
9.63.3.1	<a href="#">clicked</a>	87
9.63.3.2	<a href="#">dbValueChanged</a>	87
9.63.3.3	<a href="#">launchGui</a>	87
9.63.3.4	<a href="#">pressed</a>	87
9.63.3.5	<a href="#">released</a>	88
9.63.4	<a href="#">Property Documentation</a>	88
9.63.4.1	<a href="#">addUnits</a>	88
9.63.4.2	<a href="#">alignment</a>	88
9.63.4.3	<a href="#">allowDrop</a>	88
9.63.4.4	<a href="#">arguments</a>	88
9.63.4.5	<a href="#">arrayAction</a>	88
9.63.4.6	<a href="#">clickCheckedText</a>	89
9.63.4.7	<a href="#">clickText</a>	89
9.63.4.8	<a href="#">confirmAction</a>	89
9.63.4.9	<a href="#">creationOption</a>	89



9.63.4.10 displayAlarmState . . . . .	89
9.63.4.11 format . . . . .	90
9.63.4.12 guiFile . . . . .	90
9.63.4.13 int . . . . .	90
9.63.4.14 labelText . . . . .	90
9.63.4.15 leadingZero . . . . .	90
9.63.4.16 localEnumeration . . . . .	90
9.63.4.17 notation . . . . .	91
9.63.4.18 password . . . . .	91
9.63.4.19 pixmap0 . . . . .	91
9.63.4.20 pixmap1 . . . . .	91
9.63.4.21 pixmap2 . . . . .	92
9.63.4.22 pixmap3 . . . . .	92
9.63.4.23 pixmap4 . . . . .	92
9.63.4.24 pixmap5 . . . . .	92
9.63.4.25 pixmap6 . . . . .	92
9.63.4.26 pixmap7 . . . . .	92
9.63.4.27 precision . . . . .	92
9.63.4.28 pressText . . . . .	92
9.63.4.29 prioritySubstitutions . . . . .	93
9.63.4.30 program . . . . .	93
9.63.4.31 releaseText . . . . .	93
9.63.4.32 subscribe . . . . .	93
9.63.4.33 trailingZeros . . . . .	93
9.63.4.34 updateOption . . . . .	93
9.63.4.35 useDbPrecision . . . . .	93
9.63.4.36 userLevelEnabled . . . . .	94
9.63.4.37 userLevelEngineerStyle . . . . .	94
9.63.4.38 userLevelScientistStyle . . . . .	94
9.63.4.39 userLevelUserStyle . . . . .	94
9.63.4.40 userLevelVisibility . . . . .	94
9.63.4.41 variable . . . . .	95
9.63.4.42 variableAsToolTip . . . . .	95
9.63.4.43 variableSubstitutions . . . . .	95

9.63.4.44 visible . . . . .	95
9.63.4.45 writeOnClick . . . . .	95
9.63.4.46 writeOnPress . . . . .	95
9.63.4.47 writeOnRelease . . . . .	95
9.64 QECheckBoxManager Class Reference . . . . .	96
9.65 QEComboBox Class Reference . . . . .	96
9.65.1 Member Enumeration Documentation . . . . .	98
9.65.1.1 UserLevels . . . . .	98
9.65.2 Member Function Documentation . . . . .	98
9.65.2.1 dbValueChanged . . . . .	98
9.65.3 Member Data Documentation . . . . .	98
9.65.3.1 useDbEnumerations . . . . .	98
9.65.3.2 writeOnChange . . . . .	99
9.65.4 Property Documentation . . . . .	99
9.65.4.1 allowDrop . . . . .	99
9.65.4.2 displayAlarmState . . . . .	99
9.65.4.3 int . . . . .	99
9.65.4.4 localEnumeration . . . . .	99
9.65.4.5 subscribe . . . . .	99
9.65.4.6 userLevelEnabled . . . . .	99
9.65.4.7 userLevelEngineerStyle . . . . .	100
9.65.4.8 userLevelScientistStyle . . . . .	100
9.65.4.9 userLevelUserStyle . . . . .	100
9.65.4.10 userLevelVisibility . . . . .	100
9.65.4.11 variable . . . . .	100
9.65.4.12 variableAsToolTip . . . . .	101
9.65.4.13 variableSubstitutions . . . . .	101
9.65.4.14 visible . . . . .	101
9.66 QEConfiguredLayout Class Reference . . . . .	101
9.67 QEConfiguredLayoutManager Class Reference . . . . .	103
9.68 QEDragDrop Class Reference . . . . .	103
9.69 QEFileBrowser Class Reference . . . . .	105
9.70 QEFloating Class Reference . . . . .	106
9.71 QEFloatingFormatting Class Reference . . . . .	107

9.72	QForm Class Reference	108
9.72.1	Member Function Documentation	109
9.72.1.1	setVariableNameAndSubstitutions	109
9.73	QFrame Class Reference	109
9.73.1	Member Enumeration Documentation	111
9.73.1.1	UserLevels	111
9.73.2	Property Documentation	111
9.73.2.1	allowDrop	111
9.73.2.2	displayAlarmState	111
9.73.2.3	int	111
9.73.2.4	userLevelEnabled	111
9.73.2.5	userLevelEngineerStyle	112
9.73.2.6	userLevelScientistStyle	112
9.73.2.7	userLevelUserStyle	112
9.73.2.8	userLevelVisibility	112
9.73.2.9	variableAsToolTip	112
9.73.2.10	visible	113
9.74	QGenericButton Class Reference	113
9.75	QGenericEdit Class Reference	115
9.75.1	Member Enumeration Documentation	117
9.75.1.1	UserLevels	117
9.75.2	Constructor & Destructor Documentation	117
9.75.2.1	QGenericEdit	117
9.75.2.2	QGenericEdit	117
9.75.3	Member Function Documentation	117
9.75.3.1	getConfirmWrite	117
9.75.3.2	getSubscribe	117
9.75.3.3	getWriteOnEnter	118
9.75.3.4	getWriteOnFinish	118
9.75.3.5	getWriteOnLoseFocus	118
9.75.3.6	setConfirmWrite	118
9.75.3.7	setSubscribe	118
9.75.3.8	setWriteOnEnter	118
9.75.3.9	setWriteOnFinish	118

9.75.3.10	<a href="#">setWriteOnLoseFocus</a>	118
9.75.4	<a href="#">Property Documentation</a>	118
9.75.4.1	<a href="#">allowDrop</a>	119
9.75.4.2	<a href="#">confirmWrite</a>	119
9.75.4.3	<a href="#">displayAlarmState</a>	119
9.75.4.4	<a href="#">int</a>	119
9.75.4.5	<a href="#">subscribe</a>	119
9.75.4.6	<a href="#">userLevelEnabled</a>	119
9.75.4.7	<a href="#">userLevelEngineerStyle</a>	120
9.75.4.8	<a href="#">userLevelScientistStyle</a>	120
9.75.4.9	<a href="#">userLevelUserStyle</a>	120
9.75.4.10	<a href="#">userLevelVisibility</a>	120
9.75.4.11	<a href="#">variable</a>	120
9.75.4.12	<a href="#">variableAsToolTip</a>	120
9.75.4.13	<a href="#">variableSubstitutions</a>	121
9.75.4.14	<a href="#">visible</a>	121
9.75.4.15	<a href="#">writeOnEnter</a>	121
9.75.4.16	<a href="#">writeOnFinish</a>	121
9.75.4.17	<a href="#">writeOnLoseFocus</a>	121
9.76	<a href="#">QEGroupBox Class Reference</a>	121
9.76.1	<a href="#">Member Enumeration Documentation</a>	122
9.76.1.1	<a href="#">UserLevels</a>	122
9.76.2	<a href="#">Property Documentation</a>	123
9.76.2.1	<a href="#">allowDrop</a>	123
9.76.2.2	<a href="#">displayAlarmState</a>	123
9.76.2.3	<a href="#">int</a>	123
9.76.2.4	<a href="#">userLevelEnabled</a>	123
9.76.2.5	<a href="#">userLevelEngineerStyle</a>	123
9.76.2.6	<a href="#">userLevelScientistStyle</a>	124
9.76.2.7	<a href="#">userLevelUserStyle</a>	124
9.76.2.8	<a href="#">userLevelVisibility</a>	124
9.76.2.9	<a href="#">variableAsToolTip</a>	124
9.76.2.10	<a href="#">visible</a>	124
9.77	<a href="#">QEImage Class Reference</a>	125

9.77.1	Member Enumeration Documentation	132
9.77.1.1	formatOptions	132
9.77.1.2	FormatOptions	133
9.77.1.3	ResizeOptions	133
9.77.1.4	resizeOptions	133
9.77.1.5	rotationOptions	133
9.77.1.6	RotationOptions	133
9.77.1.7	selectOptions	134
9.77.1.8	UserLevels	134
9.77.2	Constructor & Destructor Documentation	134
9.77.2.1	QEImage	134
9.77.2.2	QEImage	135
9.77.3	Member Function Documentation	135
9.77.3.1	dbValueChanged	135
9.77.4	Member Data Documentation	135
9.77.4.1	autoBrightnessContrast	135
9.77.4.2	displayButtonBar	135
9.77.4.3	enableBrightnessContrast	135
9.77.4.4	initialVertScrollPos	135
9.77.5	Property Documentation	135
9.77.5.1	allowDrop	135
9.77.5.2	areaColor	136
9.77.5.3	beamColor	136
9.77.5.4	beamXVariable	136
9.77.5.5	beamYVariable	136
9.77.5.6	clippingHighVariable	136
9.77.5.7	clippingLowVariable	136
9.77.5.8	clippingOnOffVariable	136
9.77.5.9	displayAlarmState	136
9.77.5.10	enableHozSliceSelection	137
9.77.5.11	enableVertSliceSelection	137
9.77.5.12	formatOption	137
9.77.5.13	heightVariable	137
9.77.5.14	horizontalFlip	137

9.77.5.15	hozSliceColor	137
9.77.5.16	imageVariable	137
9.77.5.17	initialHosScrollPos	137
9.77.5.18	int	137
9.77.5.19	profileColor	138
9.77.5.20	regionOfInterest1HVariable	138
9.77.5.21	regionOfInterest1WVariable	138
9.77.5.22	regionOfInterest1XVariable	138
9.77.5.23	regionOfInterest1YVariable	138
9.77.5.24	regionOfInterest2HVariable	138
9.77.5.25	regionOfInterest2WVariable	138
9.77.5.26	regionOfInterest2XVariable	138
9.77.5.27	regionOfInterest2YVariable	138
9.77.5.28	regionOfInterest3HVariable	139
9.77.5.29	regionOfInterest3WVariable	139
9.77.5.30	regionOfInterest3XVariable	139
9.77.5.31	regionOfInterest3YVariable	139
9.77.5.32	regionOfInterest4HVariable	139
9.77.5.33	regionOfInterest4WVariable	139
9.77.5.34	regionOfInterest4XVariable	139
9.77.5.35	regionOfInterest4YVariable	139
9.77.5.36	resizeOption	139
9.77.5.37	rotation	140
9.77.5.38	showTime	140
9.77.5.39	targetColor	140
9.77.5.40	targetTriggerVariable	140
9.77.5.41	targetXVariable	140
9.77.5.42	targetYVariable	140
9.77.5.43	timeColor	140
9.77.5.44	userLevelEnabled	140
9.77.5.45	userLevelEngineerStyle	141
9.77.5.46	userLevelScientistStyle	141
9.77.5.47	userLevelUserStyle	141
9.77.5.48	userLevelVisibility	141

9.77.5.49	<a href="#">variableAsToolTip</a>	141
9.77.5.50	<a href="#">variableSubstitutions</a>	142
9.77.5.51	<a href="#">verticalFlip</a>	142
9.77.5.52	<a href="#">vertSliceColor</a>	142
9.77.5.53	<a href="#">visible</a>	142
9.77.5.54	<a href="#">widthVariable</a>	142
9.78	<a href="#">QEInteger Class Reference</a>	142
9.79	<a href="#">QEIntegerFormatting Class Reference</a>	143
9.79.1	<a href="#">Detailed Description</a>	144
9.79.2	<a href="#">Member Function Documentation</a>	144
9.79.2.1	<a href="#">formatInteger</a>	144
9.79.2.2	<a href="#">formatIntegerArray</a>	144
9.79.2.3	<a href="#">formatValue</a>	144
9.80	<a href="#">QELabel Class Reference</a>	144
9.80.1	<a href="#">Detailed Description</a>	148
9.80.2	<a href="#">Member Enumeration Documentation</a>	148
9.80.2.1	<a href="#">ArrayActions</a>	148
9.80.2.2	<a href="#">Formats</a>	148
9.80.2.3	<a href="#">Notations</a>	148
9.80.2.4	<a href="#">UpdateOptions</a>	149
9.80.2.5	<a href="#">updateOptions</a>	149
9.80.2.6	<a href="#">UserLevels</a>	149
9.80.3	<a href="#">Constructor &amp; Destructor Documentation</a>	149
9.80.3.1	<a href="#">QELabel</a>	149
9.80.3.2	<a href="#">QELabel</a>	149
9.80.4	<a href="#">Member Function Documentation</a>	150
9.80.4.1	<a href="#">dbValueChanged</a>	150
9.80.5	<a href="#">Property Documentation</a>	150
9.80.5.1	<a href="#">addUnits</a>	150
9.80.5.2	<a href="#">allowDrop</a>	150
9.80.5.3	<a href="#">arrayAction</a>	150
9.80.5.4	<a href="#">displayAlarmState</a>	150
9.80.5.5	<a href="#">format</a>	151
9.80.5.6	<a href="#">int</a>	151

9.80.5.7	leadingZero	151
9.80.5.8	localEnumeration	151
9.80.5.9	notation	152
9.80.5.10	pixmap0	152
9.80.5.11	pixmap1	152
9.80.5.12	pixmap2	152
9.80.5.13	pixmap3	152
9.80.5.14	pixmap4	152
9.80.5.15	pixmap5	152
9.80.5.16	pixmap6	152
9.80.5.17	pixmap7	153
9.80.5.18	precision	153
9.80.5.19	trailingZeros	153
9.80.5.20	updateOption	153
9.80.5.21	useDbPrecision	153
9.80.5.22	userLevelEnabled	153
9.80.5.23	userLevelEngineerStyle	153
9.80.5.24	userLevelScientistStyle	154
9.80.5.25	userLevelUserStyle	154
9.80.5.26	userLevelVisibility	154
9.80.5.27	variable	154
9.80.5.28	variableAsToolTip	154
9.80.5.29	variableSubstitutions	154
9.80.5.30	visible	155
9.81	QLineEdit Class Reference	155
9.81.1	Member Enumeration Documentation	156
9.81.1.1	ArrayActions	156
9.81.1.2	Formats	157
9.81.1.3	Notations	157
9.81.2	Constructor & Destructor Documentation	157
9.81.2.1	QLineEdit	157
9.81.2.2	QLineEdit	157
9.81.3	Member Function Documentation	157
9.81.3.1	dbValueChanged	157



9.81.4	Property Documentation	158
9.81.4.1	addUnits	158
9.81.4.2	arrayAction	158
9.81.4.3	format	158
9.81.4.4	int	158
9.81.4.5	leadingZero	158
9.81.4.6	localEnumeration	158
9.81.4.7	notation	159
9.81.4.8	precision	159
9.81.4.9	trailingZeros	159
9.81.4.10	useDbPrecision	159
9.82	QELineEditManager Class Reference	160
9.83	QELink Class Reference	160
9.84	QELocalEnumeration Class Reference	162
9.84.1	Detailed Description	162
9.84.2	Constructor & Destructor Documentation	163
9.84.2.1	QELocalEnumeration	163
9.84.2.2	QELocalEnumeration	163
9.84.3	Member Function Documentation	163
9.84.3.1	getLocalEnumeration	163
9.84.3.2	isDefined	163
9.84.3.3	setLocalEnumeration	163
9.84.3.4	textToDouble	164
9.84.3.5	textToInt	164
9.84.3.6	textToValue	164
9.84.3.7	valueToText	164
9.85	QELog Class Reference	164
9.86	QELogin Class Reference	167
9.87	QELoginDialog Class Reference	167
9.88	QENumericEdit Class Reference	168
9.88.1	Detailed Description	170
9.88.2	Constructor & Destructor Documentation	170
9.88.2.1	QENumericEdit	170
9.88.2.2	QENumericEdit	170

9.88.3	Member Function Documentation	170
9.88.3.1	dbValueChanged	170
9.88.4	Property Documentation	171
9.88.4.1	addUnits	171
9.88.4.2	autoScale	171
9.88.4.3	leadingZeros	171
9.88.4.4	maximum	171
9.88.4.5	minimum	171
9.88.4.6	precision	171
9.89	QENumericEditManager Class Reference	171
9.90	QEPeriodic Class Reference	172
9.90.1	Member Enumeration Documentation	175
9.90.1.1	UserLevels	175
9.90.2	Member Function Documentation	175
9.90.2.1	dbElementChanged	175
9.90.2.2	dbValueChanged	176
9.90.3	Member Data Documentation	176
9.90.3.1	allowDrop	176
9.90.4	Property Documentation	176
9.90.4.1	displayAlarmState	176
9.90.4.2	int	176
9.90.4.3	readbackLabelVariable1	176
9.90.4.4	readbackLabelVariable2	176
9.90.4.5	subscribe	177
9.90.4.6	userLevelEnabled	177
9.90.4.7	userLevelEngineerStyle	177
9.90.4.8	userLevelScientistStyle	177
9.90.4.9	userLevelUserStyle	177
9.90.4.10	userLevelVisibility	178
9.90.4.11	variableAsToolTip	178
9.90.4.12	variableSubstitutions	178
9.90.4.13	visible	178
9.90.4.14	writeButtonVariable1	178
9.90.4.15	writeButtonVariable2	178

9.91	QEP periodicComponentData Class Reference	178
9.92	QEP periodicTaskMenu Class Reference	179
9.93	QEP periodicTaskMenuFactory Class Reference	179
9.94	QEPicsPV Class Reference	180
9.95	QEPPlot Class Reference	181
9.95.1	Member Enumeration Documentation	184
9.95.1.1	UserLevels	184
9.95.2	Member Function Documentation	184
9.95.2.1	dbValueChanged	184
9.95.2.2	dbValueChanged	184
9.95.3	Member Data Documentation	185
9.95.3.1	allowDrop	185
9.95.4	Property Documentation	185
9.95.4.1	displayAlarmState	185
9.95.4.2	int	185
9.95.4.3	userLevelEnabled	185
9.95.4.4	userLevelEngineerStyle	185
9.95.4.5	userLevelScientistStyle	186
9.95.4.6	userLevelUserStyle	186
9.95.4.7	userLevelVisibility	186
9.95.4.8	variable1	186
9.95.4.9	variable2	186
9.95.4.10	variable3	186
9.95.4.11	variable4	187
9.95.4.12	variableAsToolTip	187
9.95.4.13	variableSubstitutions	187
9.95.4.14	visible	187
9.96	QEPushButton Class Reference	187
9.96.1	Member Enumeration Documentation	190
9.96.1.1	ArrayActions	190
9.96.1.2	CreationOptionNames	191
9.96.1.3	Formats	191
9.96.1.4	Notations	191
9.96.1.5	UpdateOptions	191

9.96.1.6	UserLevels	192
9.96.2	Constructor & Destructor Documentation	192
9.96.2.1	QEPushButton	192
9.96.2.2	QEPushButton	192
9.96.3	Member Function Documentation	192
9.96.3.1	clicked	192
9.96.3.2	dbValueChanged	192
9.96.3.3	launchGui	192
9.96.3.4	pressed	193
9.96.3.5	released	193
9.96.4	Property Documentation	193
9.96.4.1	addUnits	193
9.96.4.2	alignment	193
9.96.4.3	allowDrop	193
9.96.4.4	altReadbackVariable	193
9.96.4.5	arguments	193
9.96.4.6	arrayAction	194
9.96.4.7	clickCheckedText	194
9.96.4.8	clickText	194
9.96.4.9	confirmAction	194
9.96.4.10	creationOption	195
9.96.4.11	displayAlarmState	195
9.96.4.12	format	195
9.96.4.13	guiFile	195
9.96.4.14	int	195
9.96.4.15	labelText	196
9.96.4.16	leadingZero	196
9.96.4.17	localEnumeration	196
9.96.4.18	notation	197
9.96.4.19	password	197
9.96.4.20	pixmap0	197
9.96.4.21	pixmap1	197
9.96.4.22	pixmap2	197
9.96.4.23	pixmap3	197

9.96.4.24 pixmap4 . . . . .	197
9.96.4.25 pixmap5 . . . . .	197
9.96.4.26 pixmap6 . . . . .	197
9.96.4.27 pixmap7 . . . . .	198
9.96.4.28 precision . . . . .	198
9.96.4.29 pressText . . . . .	198
9.96.4.30 prioritySubstitutions . . . . .	198
9.96.4.31 program . . . . .	198
9.96.4.32 releaseText . . . . .	198
9.96.4.33 subscribe . . . . .	198
9.96.4.34 trailingZeros . . . . .	199
9.96.4.35 updateOption . . . . .	199
9.96.4.36 useDbPrecision . . . . .	199
9.96.4.37 userLevelEnabled . . . . .	199
9.96.4.38 userLevelEngineerStyle . . . . .	199
9.96.4.39 userLevelScientistStyle . . . . .	199
9.96.4.40 userLevelUserStyle . . . . .	200
9.96.4.41 userLevelVisibility . . . . .	200
9.96.4.42 variable . . . . .	200
9.96.4.43 variableAsToolTip . . . . .	200
9.96.4.44 variableSubstitutions . . . . .	200
9.96.4.45 visible . . . . .	200
9.96.4.46 writeOnClick . . . . .	200
9.96.4.47 writeOnPress . . . . .	201
9.96.4.48 writeOnRelease . . . . .	201
9.97 QEPVNameLists Class Reference . . . . .	201
9.98 QEPvProperties Class Reference . . . . .	201
9.98.1 Member Enumeration Documentation . . . . .	203
9.98.1.1 UserLevels . . . . .	203
9.98.2 Member Function Documentation . . . . .	203
9.98.2.1 restoreConfiguration . . . . .	203
9.98.2.2 saveConfiguration . . . . .	203
9.98.2.3 scaleBy . . . . .	204
9.98.3 Property Documentation . . . . .	204

9.98.3.1	<a href="#">allowDrop</a>	204
9.98.3.2	<a href="#">displayAlarmState</a>	204
9.98.3.3	<a href="#">int</a>	204
9.98.3.4	<a href="#">userLevelEnabled</a>	204
9.98.3.5	<a href="#">userLevelEngineerStyle</a>	205
9.98.3.6	<a href="#">userLevelScientistStyle</a>	205
9.98.3.7	<a href="#">userLevelUserStyle</a>	205
9.98.3.8	<a href="#">userLevelVisibility</a>	205
9.98.3.9	<a href="#">variable</a>	205
9.98.3.10	<a href="#">variableAsToolTip</a>	206
9.98.3.11	<a href="#">variableSubstitutions</a>	206
9.98.3.12	<a href="#">visible</a>	206
9.99	<a href="#">QEPvPropertiesManager Class Reference</a>	206
9.100	<a href="#">QERadioButton Class Reference</a>	207
9.100.1	<a href="#">Member Enumeration Documentation</a>	210
9.100.1.1	<a href="#">ArrayActions</a>	210
9.100.1.2	<a href="#">CreationOptionNames</a>	210
9.100.1.3	<a href="#">Formats</a>	210
9.100.1.4	<a href="#">Notations</a>	210
9.100.1.5	<a href="#">UpdateOptions</a>	211
9.100.1.6	<a href="#">UserLevels</a>	211
9.100.2	<a href="#">Constructor &amp; Destructor Documentation</a>	211
9.100.2.1	<a href="#">QERadioButton</a>	211
9.100.2.2	<a href="#">QERadioButton</a>	211
9.100.3	<a href="#">Member Function Documentation</a>	211
9.100.3.1	<a href="#">clicked</a>	211
9.100.3.2	<a href="#">dbValueChanged</a>	212
9.100.3.3	<a href="#">launchGui</a>	212
9.100.3.4	<a href="#">pressed</a>	212
9.100.3.5	<a href="#">released</a>	212
9.100.4	<a href="#">Property Documentation</a>	212
9.100.4.1	<a href="#">addUnits</a>	212
9.100.4.2	<a href="#">alignment</a>	212
9.100.4.3	<a href="#">allowDrop</a>	213

9.100.4.4 arguments	213
9.100.4.5 arrayAction	213
9.100.4.6 clickCheckedText	213
9.100.4.7 clickText	213
9.100.4.8 confirmAction	214
9.100.4.9 creationOption	214
9.100.4.10displayAlarmState	214
9.100.4.11format	214
9.100.4.12guiFile	214
9.100.4.13nt	214
9.100.4.14labelText	215
9.100.4.15leadingZero	215
9.100.4.16localEnumeration	215
9.100.4.17notation	216
9.100.4.18password	216
9.100.4.19pixmap0	216
9.100.4.20pixmap1	216
9.100.4.21pixmap2	216
9.100.4.22pixmap3	216
9.100.4.23pixmap4	216
9.100.4.24pixmap5	216
9.100.4.25pixmap6	217
9.100.4.26pixmap7	217
9.100.4.27precision	217
9.100.4.28pressText	217
9.100.4.29prioritySubstitutions	217
9.100.4.30program	217
9.100.4.31releaseText	217
9.100.4.32subscribe	218
9.100.4.33trailingZeros	218
9.100.4.34updateOption	218
9.100.4.35useDbPrecision	218
9.100.4.36userLevelEnabled	218
9.100.4.37userLevelEngineerStyle	218

9.100.4.38userLevelScientistStyle . . . . .	218
9.100.4.39userLevelUserStyle . . . . .	219
9.100.4.40userLevelVisibility . . . . .	219
9.100.4.41variable . . . . .	219
9.100.4.42variableAsToolTip . . . . .	219
9.100.4.43variableSubstitutions . . . . .	219
9.100.4.44visible . . . . .	219
9.100.4.45writeOnClick . . . . .	220
9.100.4.46writeOnPress . . . . .	220
9.100.4.47writeOnRelease . . . . .	220
9.101QERecipe Class Reference . . . . .	220
9.102QERecordFieldName Class Reference . . . . .	222
9.103QERecordSpec Class Reference . . . . .	223
9.104QERecordSpecList Class Reference . . . . .	223
9.105QEScript Class Reference . . . . .	223
9.106QEShape Class Reference . . . . .	225
9.106.1 Detailed Description . . . . .	229
9.106.2 Member Enumeration Documentation . . . . .	229
9.106.2.1 animationOptions . . . . .	229
9.106.2.2 shapeOptions . . . . .	229
9.106.2.3 UserLevels . . . . .	229
9.106.3 Constructor & Destructor Documentation . . . . .	230
9.106.3.1 QEShape . . . . .	230
9.106.3.2 QEShape . . . . .	230
9.106.4 Member Function Documentation . . . . .	230
9.106.4.1 dbValueChanged1 . . . . .	230
9.106.4.2 dbValueChanged2 . . . . .	230
9.106.4.3 dbValueChanged3 . . . . .	230
9.106.4.4 dbValueChanged4 . . . . .	230
9.106.4.5 dbValueChanged5 . . . . .	231
9.106.4.6 dbValueChanged6 . . . . .	231
9.106.5 Property Documentation . . . . .	231
9.106.5.1 allowDrop . . . . .	231
9.106.5.2 animation1 . . . . .	231



9.106.5.3 animation2 . . . . .	231
9.106.5.4 animation3 . . . . .	231
9.106.5.5 animation4 . . . . .	231
9.106.5.6 animation5 . . . . .	231
9.106.5.7 animation6 . . . . .	232
9.106.5.8 color1 . . . . .	232
9.106.5.9 color10 . . . . .	232
9.106.5.10color2 . . . . .	232
9.106.5.11color3 . . . . .	232
9.106.5.12color4 . . . . .	232
9.106.5.13color5 . . . . .	232
9.106.5.14color6 . . . . .	232
9.106.5.15color7 . . . . .	232
9.106.5.16color8 . . . . .	233
9.106.5.17color9 . . . . .	233
9.106.5.18displayAlarmState . . . . .	233
9.106.5.19int . . . . .	233
9.106.5.20offset1 . . . . .	233
9.106.5.21offset2 . . . . .	233
9.106.5.22offset3 . . . . .	233
9.106.5.23offset4 . . . . .	234
9.106.5.24offset5 . . . . .	234
9.106.5.25offset6 . . . . .	234
9.106.5.26point1 . . . . .	234
9.106.5.27point10 . . . . .	234
9.106.5.28point2 . . . . .	234
9.106.5.29point3 . . . . .	234
9.106.5.30point4 . . . . .	234
9.106.5.31point5 . . . . .	234
9.106.5.32point6 . . . . .	235
9.106.5.33point7 . . . . .	235
9.106.5.34point8 . . . . .	235
9.106.5.35point9 . . . . .	235
9.106.5.36scale2 . . . . .	235

9.106.5.37scale3	235
9.106.5.38scale4	235
9.106.5.39scale5	235
9.106.5.40scale6	235
9.106.5.41userLevelEnabled	236
9.106.5.42userLevelEngineerStyle	236
9.106.5.43userLevelScientistStyle	236
9.106.5.44userLevelUserStyle	236
9.106.5.45userLevelVisibility	236
9.106.5.46variable1	237
9.106.5.47variable2	237
9.106.5.48variable3	237
9.106.5.49variable4	237
9.106.5.50variable5	237
9.106.5.51variable6	237
9.106.5.52variableAsToolTip	237
9.106.5.53variableSubstitutions	238
9.106.5.54visible	238
9.107QESlider Class Reference	238
9.107.1 Member Enumeration Documentation	240
9.107.1.1 UserLevels	240
9.107.2 Member Function Documentation	240
9.107.2.1 dbValueChanged	240
9.107.3 Member Data Documentation	240
9.107.3.1 writeOnChange	240
9.107.4 Property Documentation	240
9.107.4.1 allowDrop	240
9.107.4.2 displayAlarmState	240
9.107.4.3 int	241
9.107.4.4 subscribe	241
9.107.4.5 userLevelEnabled	241
9.107.4.6 userLevelEngineerStyle	241
9.107.4.7 userLevelScientistStyle	241
9.107.4.8 userLevelUserStyle	241

9.107.4.9 userLevelVisibility . . . . .	242
9.107.4.10variable . . . . .	242
9.107.4.11variableAsToolTip . . . . .	242
9.107.4.12variableSubstitutions . . . . .	242
9.107.4.13visible . . . . .	242
9.108QESpinBox Class Reference . . . . .	243
9.108.1 Member Enumeration Documentation . . . . .	244
9.108.1.1 UserLevels . . . . .	244
9.108.2 Member Function Documentation . . . . .	245
9.108.2.1 dbValueChanged . . . . .	245
9.108.3 Property Documentation . . . . .	245
9.108.3.1 allowDrop . . . . .	245
9.108.3.2 displayAlarmState . . . . .	245
9.108.3.3 int . . . . .	245
9.108.3.4 subscribe . . . . .	245
9.108.3.5 userLevelEnabled . . . . .	246
9.108.3.6 userLevelEngineerStyle . . . . .	246
9.108.3.7 userLevelScientistStyle . . . . .	246
9.108.3.8 userLevelUserStyle . . . . .	246
9.108.3.9 userLevelVisibility . . . . .	246
9.108.3.10variable . . . . .	247
9.108.3.11variableAsToolTip . . . . .	247
9.108.3.12variableSubstitutions . . . . .	247
9.108.3.13visible . . . . .	247
9.109QString Class Reference . . . . .	247
9.110QStringFormatting Class Reference . . . . .	248
9.110.1 Member Enumeration Documentation . . . . .	249
9.110.1.1 arrayActions . . . . .	249
9.110.1.2 formats . . . . .	249
9.110.1.3 notations . . . . .	250
9.111QStringFormattingMethods Class Reference . . . . .	250
9.112QEStripChart Class Reference . . . . .	251
9.112.1 Member Function Documentation . . . . .	253
9.112.1.1 restoreConfiguration . . . . .	253

9.112.1.2 saveConfiguration . . . . .	253
9.112.2 Property Documentation . . . . .	253
9.112.2.1 variableSubstitutions . . . . .	253
9.113QEStripChartAdjustPVDialo Class Reference . . . . .	254
9.114QEStripChartContextMenu Class Reference . . . . .	254
9.114.1 Constructor & Destructor Documentation . . . . .	254
9.114.1.1 QEStripChartContextMenu . . . . .	254
9.115QEStripChartItem Class Reference . . . . .	255
9.116QEStripChartItemDialog Class Reference . . . . .	256
9.117QEStripChartNames Class Reference . . . . .	256
9.118QEStripChartRangeDialog Class Reference . . . . .	257
9.119QEStripChartTimeDialog Class Reference . . . . .	258
9.120QEStripChartToolBar Class Reference . . . . .	258
9.120.1 Detailed Description . . . . .	259
9.121QESubstitutedLabel Class Reference . . . . .	259
9.121.1 Member Data Documentation . . . . .	260
9.121.1.1 labelText . . . . .	260
9.121.2 Property Documentation . . . . .	260
9.121.2.1 textSubstitutions . . . . .	260
9.122QEToolTip Class Reference . . . . .	260
9.123QEWidget Class Reference . . . . .	262
9.123.1 Detailed Description . . . . .	263
9.123.2 Member Function Documentation . . . . .	265
9.123.2.1 activate . . . . .	265
9.123.2.2 deactivate . . . . .	265
9.123.2.3 defaultFileLocation . . . . .	265
9.123.2.4 findQEFile . . . . .	265
9.123.2.5 getColor . . . . .	265
9.123.2.6 getFrameworkVersion . . . . .	265
9.123.2.7 getMessageSourceId . . . . .	265
9.123.2.8 getQcaltem . . . . .	265
9.123.2.9 openQEFile . . . . .	266
9.123.2.10processAlarmInfo . . . . .	266
9.123.2.11readNow . . . . .	266

9.123.2.12restoreConfiguration . . . . .	266
9.123.2.13saveConfiguration . . . . .	266
9.123.2.14scaleBy . . . . .	266
9.123.2.15setMessageSourceId . . . . .	267
9.123.2.16setupContextMenu . . . . .	267
9.123.2.17setVariableNameAndSubstitutions . . . . .	267
9.123.2.18writeNow . . . . .	267
9.124QEWidgets Class Reference . . . . .	267
9.125QLabelList Class Reference . . . . .	268
9.126qcastatemachine::ReadQCaStateMachine Class Reference . . . . .	268
9.127ROIInfo Class Reference . . . . .	268
9.128SaveRestoreSignal Class Reference . . . . .	269
9.128.1 Member Function Documentation . . . . .	269
9.128.1.1 restore . . . . .	269
9.128.1.2 save . . . . .	269
9.129saveRestoreSlot Class Reference . . . . .	269
9.130selectMenu Class Reference . . . . .	270
9.131standardProperties Class Reference . . . . .	270
9.132StateMachineTemplate Class Reference . . . . .	272
9.133qcastatemachine::SubscriptionQCaStateMachine Class Reference . . . . .	272
9.134trace Class Reference . . . . .	273
9.135TrackRange Class Reference . . . . .	273
9.136UserInfoStruct Class Reference . . . . .	274
9.137QEPeriodic::userInfoStructArray Struct Reference . . . . .	274
9.138userLevelSignal Class Reference . . . . .	274
9.139userLevelSlot Class Reference . . . . .	275
9.140userLevelTypes Class Reference . . . . .	275
9.140.1 Member Enumeration Documentation . . . . .	275
9.140.1.1 userLevels . . . . .	275
9.141UserMessage Class Reference . . . . .	275
9.141.1 Detailed Description . . . . .	277
9.142UserMessageSignal Class Reference . . . . .	278
9.142.1 Detailed Description . . . . .	279
9.143UserMessageSlot Class Reference . . . . .	279

9.143.1 Detailed Description . . . . .	279
9.144ValueScaling Class Reference . . . . .	280
9.145VideoWidget Class Reference . . . . .	280
9.146WidgetRef Class Reference . . . . .	281
9.147qcastatemachine::WriteQCaStateMachine Class Reference . . . . .	281
9.148zoomMenu Class Reference . . . . .	282

## 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  $\geq 4.4$ .

## 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
```

Alternatively, 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 environment for building or using the epicsqt framework.

To build the framework, open epicsqt.pro in QtCreator, ensure shadow 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](mailto: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

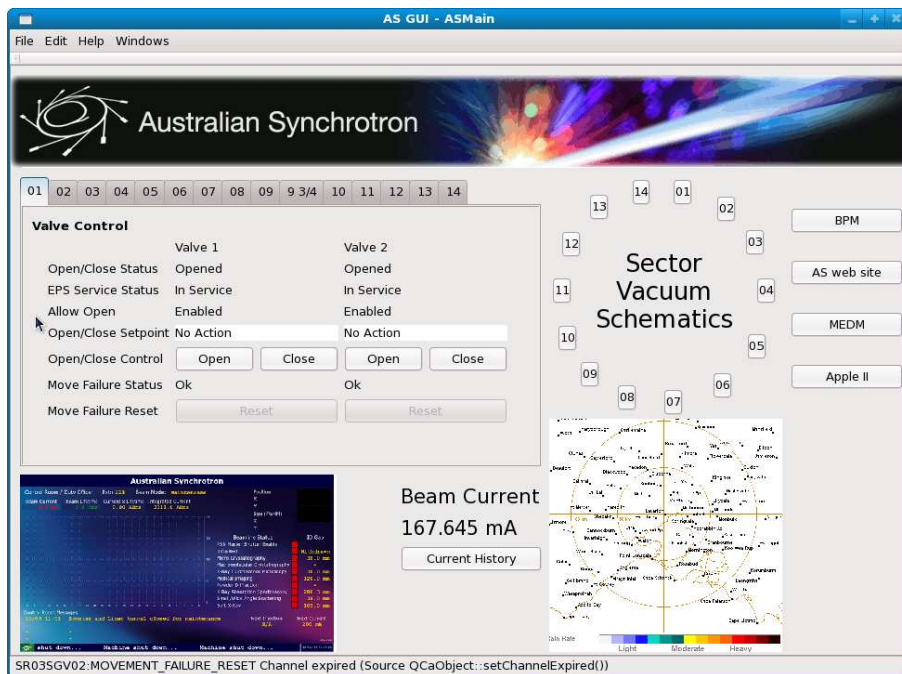


Figure 3.1: Australian Synchrotron mock up

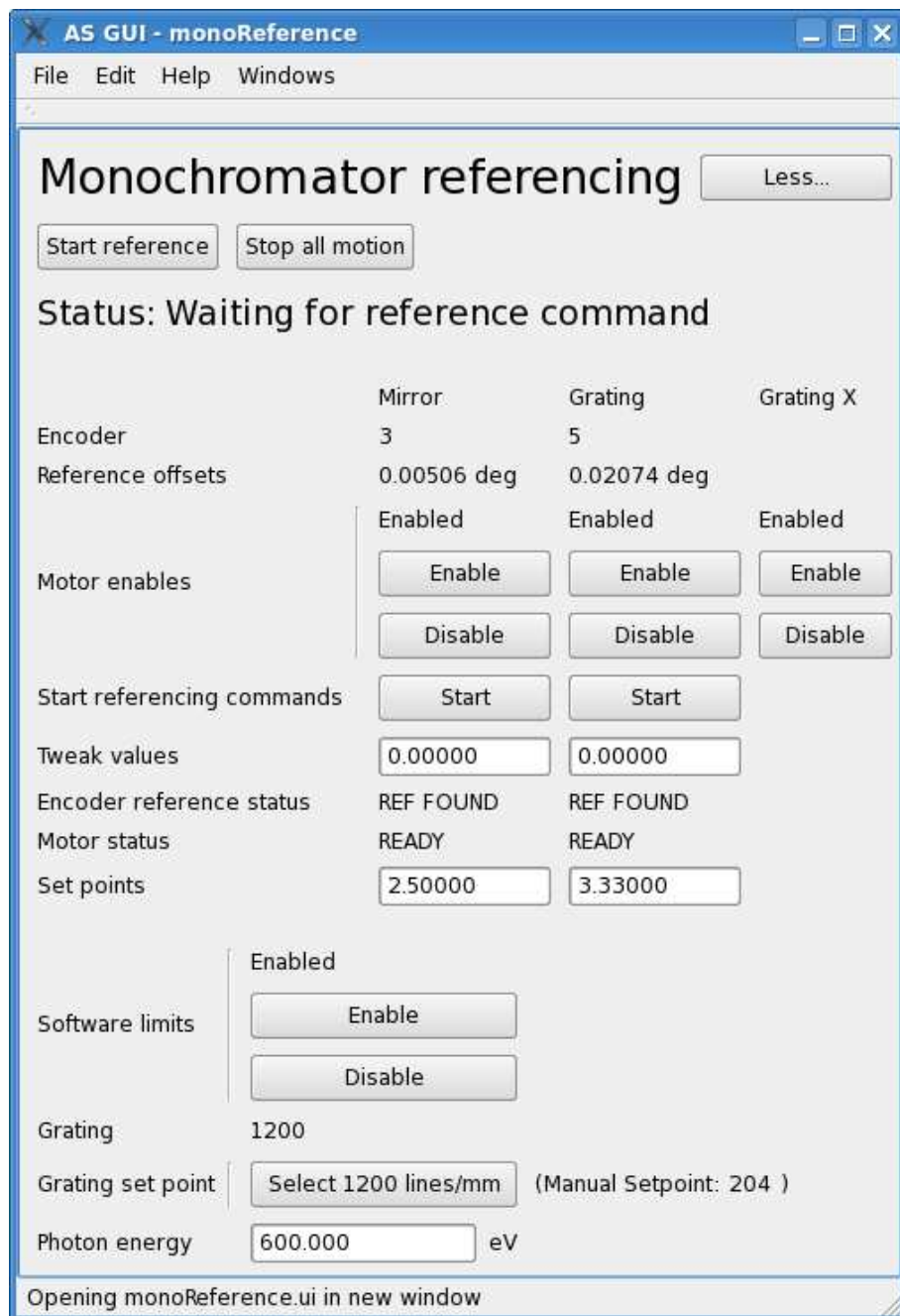


Figure 3.2: Monochromator referencing

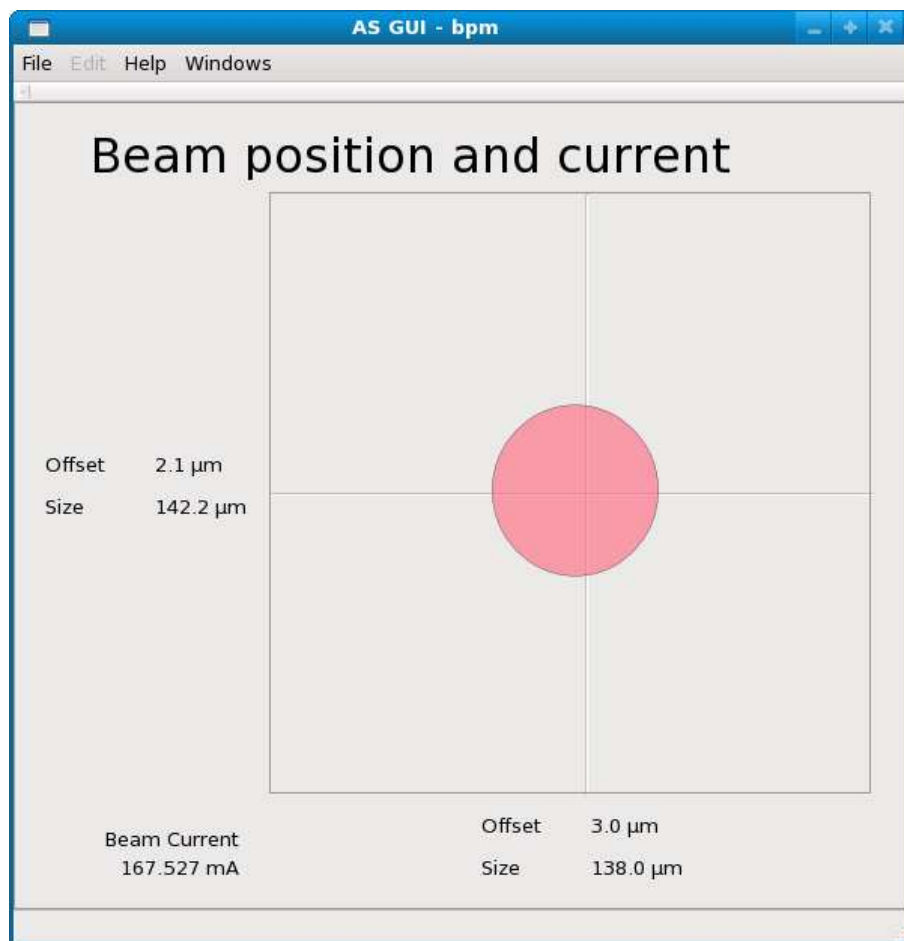


Figure 3.3: Beam position monitor

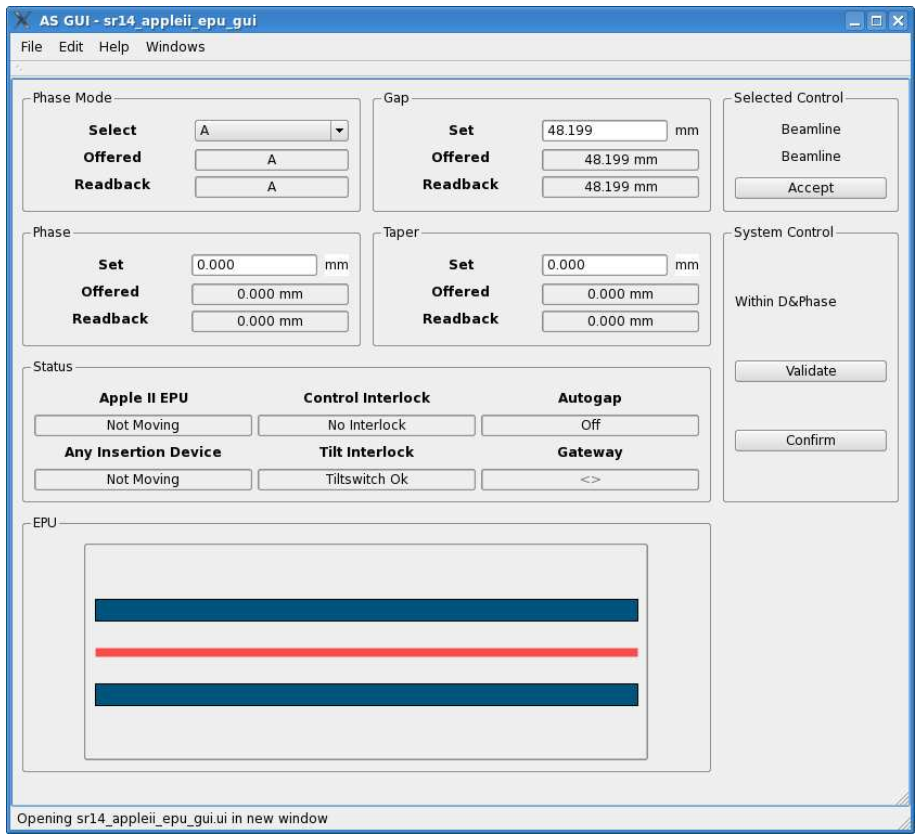


Figure 3.4: Insertion device





Figure 3.5: Injection efficiency monitor



## Chapter 4

# other applications using epicsqt widgets

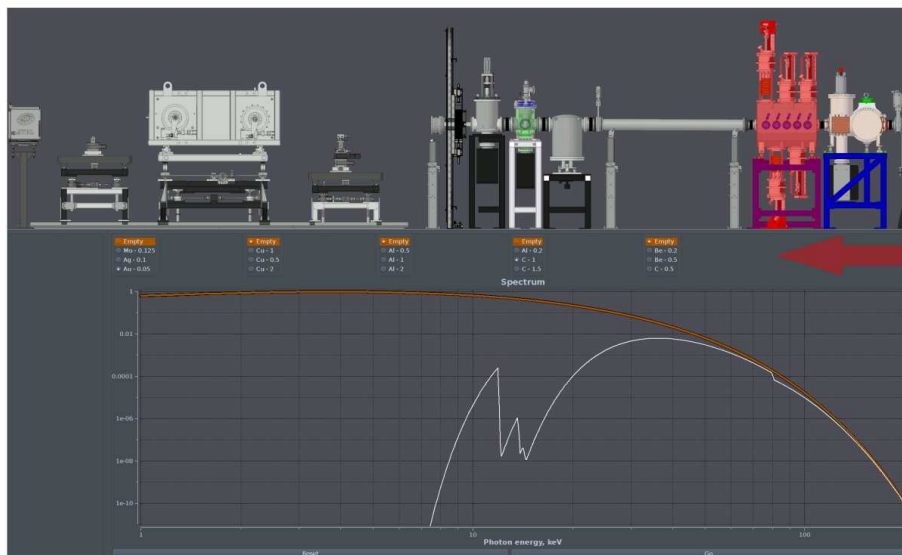


Figure 4.1: Medical Imaging beamline

2B SampleTable Z <@SR08ID01OPI01>

PV name: SR08ID01:MTR32B View mode: Macro

Description: 2B SampleTable Z

Precision: 5 Units: mm

Message: Connection established. clean

User: 6mm Move absolutely Raw: -80932

JOG< UNDO >JOG

LIMIT< Move relatively >LIMIT

< 1mm >

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

User: 6mm = Hi limit

Resolution: 0.0001mm/step \* 42271.2068mm

Raw: -80932 + Lo Limit

Offset: -2.0932mm -42275.3932mm

Speed Acceleration

Maximum: 1.2mm/s

Normal: 0.8mm/s 1s

Backlash: 0mm/s 1s

log: 1mm/s 10s

Backlash: 0mm

Figure 4.2: Motor controller

MotorMx <@SR08ID01OPI01>

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

Add motor

Figure 4.3: Motor controller

## Chapter 5

# Qt Designer

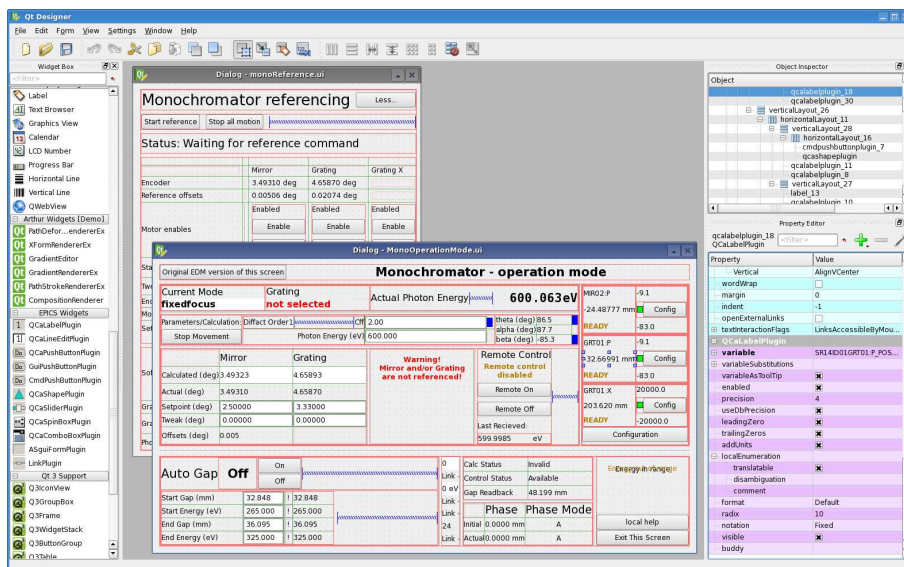


Figure 5.1: Editing multiple GUIs

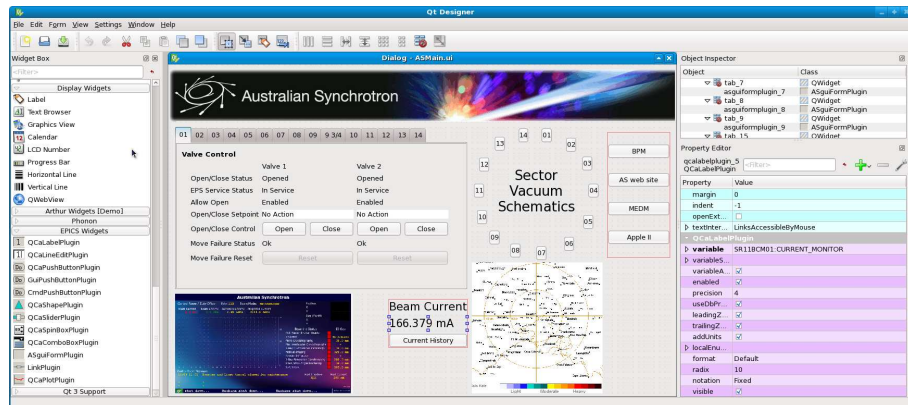


Figure 5.2: Editing a GUI

## Chapter 6

# Qt Creator

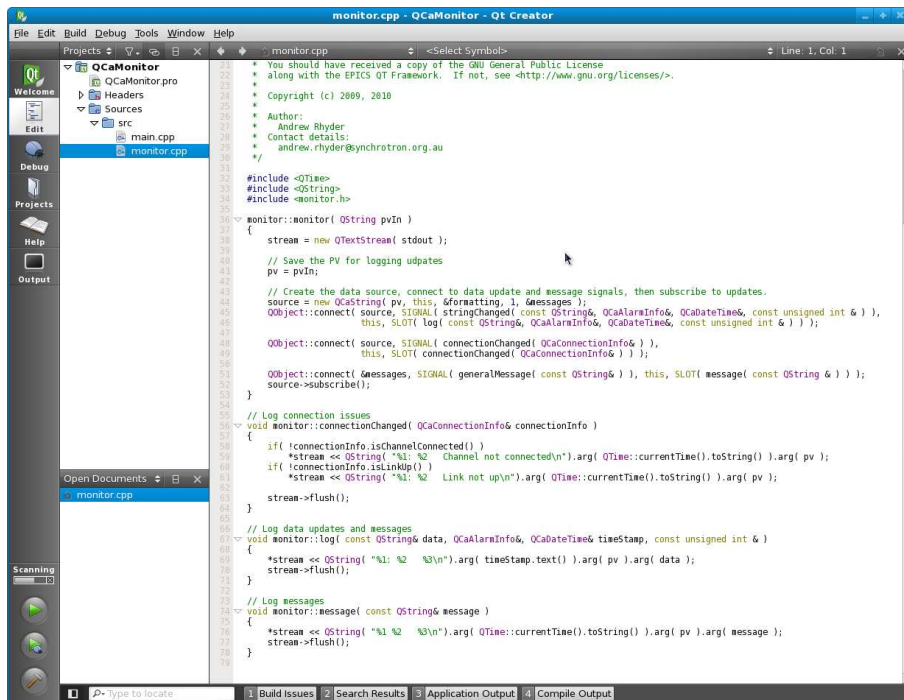


Figure 6.1: Application using epicsqt data source classes





## Chapter 7

# Class Index

### 7.1 Class Hierarchy

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

_Field . . . . .	27
_Item . . . . .	28
_QDialogItem . . . . .	28
_QPushButtonGroup . . . . .	28
_QTableWidgetFileBrowser . . . . .	29
_QTableWidgetLog . . . . .	29
_QTableWidgetScript . . . . .	29
QEAAnalogIndicator::Band . . . . .	30
QEAAnalogIndicator::BandList . . . . .	30
ChartState . . . . .	30
ContainerProfile . . . . .	31
QEWidget . . . . .	262
QEAAnalogProgressBar . . . . .	68
QEBitStatus . . . . .	77
QEComboBox . . . . .	96
QEConfiguredLayout . . . . .	101
QEFileBrowser . . . . .	105
QEForm . . . . .	108
QEFrame . . . . .	109
QEGenericButton . . . . .	113
QECheckBox . . . . .	82
QEPushButton . . . . .	187
QERadioButton . . . . .	207
QEGenericEdit . . . . .	115
QELineEdit . . . . .	155
QENumericEdit . . . . .	168
QEGroupBox . . . . .	121
QEImage . . . . .	125
QELabel . . . . .	144

QELink . . . . .	160
QELog . . . . .	164
QELogin . . . . .	167
QEPeiodic . . . . .	172
QEPlot . . . . .	181
QEPvProperties . . . . .	201
QERecipe . . . . .	220
QEScript . . . . .	223
QEShape . . . . .	225
QESlider . . . . .	238
QESpinBox . . . . .	243
QEStripChart . . . . .	251
QESubstitutedLabel . . . . .	259
contextMenu . . . . .	33
QEWidget . . . . .	262
contextMenuObject . . . . .	35
QEPeiodic::elementInfoStruct . . . . .	35
flipRotateMenu . . . . .	36
imageContextMenu . . . . .	36
imageInfo . . . . .	37
QEImage . . . . .	125
imageMarkup . . . . .	38
VideoWidget . . . . .	280
loginWidget . . . . .	39
managePixmaps . . . . .	40
QEGenericButton . . . . .	113
QELabel . . . . .	144
markupItem . . . . .	42
markupBeam . . . . .	40
markupHLine . . . . .	41
markupLine . . . . .	44
markupRegion . . . . .	45
markupTarget . . . . .	45
markupText . . . . .	46
markupVLine . . . . .	47
message_types . . . . .	48
QEStripChartToolBar::OwnWidgets . . . . .	48
QEPvProperties::OwnWidgets . . . . .	49
PeriodicDialog . . . . .	49
PeriodicElementSetupForm . . . . .	50
PeriodicSetupDialog . . . . .	50
PersistanceManager . . . . .	50
PMContext . . . . .	51
PMElement . . . . .	51
PMElementList . . . . .	52
QEStripChart::PrivateData . . . . .	52
QEStripChartItem::PrivateData . . . . .	53
profilePlot . . . . .	53

PublishedProfile . . . . .	54
PushButtonSpecifications . . . . .	54
QBitStatus . . . . .	55
QEBitStatus . . . . .	77
QCaAlarmInfo . . . . .	56
QCaConnectionInfo . . . . .	57
QCaDataPoint . . . . .	57
QCaDataPointList . . . . .	58
QCaDateTime . . . . .	58
QCaEventFilter . . . . .	58
QCaEventItem . . . . .	59
QCaEventUpdate . . . . .	59
QCaInstalledFiltersListItem . . . . .	60
qcaobject::QCaObject . . . . .	60
QEByteArray . . . . .	81
QEFloating . . . . .	106
QEInteger . . . . .	142
QEString . . . . .	247
QCaVariableNamePropertyManager . . . . .	62
QEAnalogIndicator . . . . .	63
QEAnalogProgressBar . . . . .	68
QEChartStateLists . . . . .	82
QECheckBoxManager . . . . .	96
QEConfiguredLayoutManager . . . . .	103
QEDragDrop . . . . .	103
QEWidget . . . . .	262
QEFloatingFormatting . . . . .	107
QEIntegerFormatting . . . . .	143
QELineEditManager . . . . .	160
QELocalEnumeration . . . . .	162
QELoginDialog . . . . .	167
QENumericEditManager . . . . .	171
QEPeriodicComponentData . . . . .	178
QEPeriodicTaskMenu . . . . .	179
QEPeriodicTaskMenuFactory . . . . .	179
QEpicsPV . . . . .	180
QEPVNameLists . . . . .	201
QEPvPropertiesManager . . . . .	206
QERecordFieldName . . . . .	222
QERecordSpec . . . . .	223
QERecordSpecList . . . . .	223
QEStringFormatting . . . . .	248
QEStringFormattingMethods . . . . .	250
QEAnalogProgressBar . . . . .	68
QEGenericButton . . . . .	113
QELabel . . . . .	144
QELineEdit . . . . .	155
QEStripChartAdjustPVDialo . . . . .	254

QEStripChartContextMenu	254
QEStripChartItem	255
QEStripChartItemDialog	256
QEStripChartNames	256
QEStripChartRangeDialog	257
QEStripChartTimeDialog	258
QEStripChartToolBar	258
QEToolTip	260
QEWidget	262
QEWidgets	267
QLabelList	268
ROInfo	268
SaveRestoreSignal	269
saveRestoreSlot	269
selectMenu	270
standardProperties	270
QEWidget	262
StateMachineTemplate	272
qcastatemachine::QCaStateMachine	62
qcastatemachine::ConnectionQCaStateMachine	31
qcastatemachine::ReadQCaStateMachine	268
qcastatemachine::SubscriptionQCaStateMachine	272
qcastatemachine::WriteQCaStateMachine	281
trace	273
TrackRange	273
userInfoStruct	274
QEPeiodic::userInfoStructArray	274
userLevelSignal	274
userLevelSlot	275
userLevelTypes	275
UserMessage	275
QEWidget	262
UserMessageSignal	278
UserMessageSlot	279
ValueScaling	280
WidgetRef	281
zoomMenu	282

## Chapter 8

# Class Index

### 8.1 Class List

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

<a href="#">_Field</a>	27
<a href="#">_Item</a>	28
<a href="#">_QDialogItem</a>	28
<a href="#">_QPushButtonGroup</a>	28
<a href="#">_QTableWidgetFileBrowser</a>	29
<a href="#">_QTableWidgetLog</a>	29
<a href="#">_QTableWidgetScript</a>	29
<a href="#">QEAAnalogIndicator::Band</a>	30
<a href="#">QEAAnalogIndicator::BandList</a>	30
<a href="#">ChartState</a>	30
<a href="#">qcastatemachine::ConnectionQCaStateMachine</a>	31
<a href="#">ContainerProfile</a>	31
<a href="#">contextMenu</a>	33
<a href="#">contextMenuObject</a>	35
<a href="#">QEPeriodic::elementInfoStruct</a>	35
<a href="#">flipRotateMenu</a>	36
<a href="#">imageContextMenu</a>	36
<a href="#">imageInfo</a>	37
<a href="#">imageMarkup</a>	38
<a href="#">loginWidget</a>	39
<a href="#">managePixmap</a>	40
<a href="#">markupBeam</a>	40
<a href="#">markupHLine</a>	41
<a href="#">markupItem</a>	42
<a href="#">markupLine</a>	44
<a href="#">markupRegion</a>	45
<a href="#">markupTarget</a>	45
<a href="#">markupText</a>	46
<a href="#">markupVLine</a>	47

message_types	48
QEStripChartToolBar::OwnWidgets	48
QEPvProperties::OwnWidgets	49
PeriodicDialog	49
PeriodicElementSetupForm	50
PeriodicSetupDialog	50
PersistenceManager	50
PMContext	51
PMElement	51
PMElementList	52
QEStripChart::PrivateData	52
QEStripChartItem::PrivateData	53
profilePlot	53
PublishedProfile	54
PushButtonSpecifications	54
QBitStatus	55
QCaAlarmInfo	56
QCaConnectionInfo	57
QCaDataPoint	57
QCaDataPointList	58
QCaDateTime	58
QCaEventFilter	58
QCaEventItem	59
QCaEventUpdate	59
QCaInstalledFiltersListItem	60
qcaobject::QCaObject	60
qcastatemachine::QCaStateMachine	62
QCaVariableNamePropertyManager	62
QEAAnalogIndicator	63
QEAAnalogProgressBar	68
QEBitStatus	77
QEByteArray	81
QEChartStateLists	82
QECheckBox	82
QECheckBoxManager	96
QEComboBox	96
QEConfiguredLayout	101
QEConfiguredLayoutManager	103
QEDragDrop	103
QEFileBrowser	105
QEFloating	106
QEFloatingFormatting	107
QEForm	108
QEFrame	109
QEGenericButton	113
QEGenericEdit	115
QEGroupBox	121
QEImage	125
QEInteger	142
QEIntegerFormatting	143

QELabel	144
QELineEdit	155
QELineEditManager	160
QELink	160
QELocalEnumeration	162
QELog	164
QELogin	167
QELoginDialog	167
QENumericEdit (The <a href="#">QENumericEdit</a> class This class is similar to <a href="#">QELineEdit</a> (both of which are derived from <a href="#">QLineEdit</a> ). However this class is tailored specifically for editing numerical values )	168
QENumericEditManager	171
QEPeriodic	172
QEPeriodicComponentData	178
QEPeriodicTaskMenu	179
QEPeriodicTaskMenuFactory	179
QEpicsPV	180
QEPlot	181
QEPushButton	187
QEPVNameLists	201
QEPvProperties	201
QEPvPropertiesManager	206
QERadioButton	207
QERecipe	220
QERecordFieldName	222
QERecordSpec	223
QERecordSpecList	223
QEScript	223
QEShape	225
QESlider	238
QESpinBox	243
QEStrng	247
QEStrngFormatting	248
QEStrngFormattingMethods	250
QEStrngChart	251
QEStrngChartAdjustPVDialg	254
QEStrngChartContextMenu	254
QEStrngChartItem	255
QEStrngChartItemDialog	256
QEStrngChartNames	256
QEStrngChartRangeDialog	257
QEStrngChartTimeDialog	258
QEStrngChartToolBar (This class holds all the StripChart tool bar widgets )	258
QESubstitutedLabel	259
QEToolTip	260
QEWidgert	262
QEWidgerts	267
QLabelList	268
qcastatemachine::ReadQCaStateMachine	268
ROlinfo	268

SaveRestoreSignal	269
saveRestoreSlot	269
selectMenu	270
standardProperties	270
StateMachineTemplate	272
qcastatemachine::SubscriptionQCaStateMachine	272
trace	273
TrackRange	273
userInfoStruct	274
QEPeiodic::userInfoStructArray	274
userLevelSignal	274
userLevelSlot	275
userLevelTypes	275
UserMessage	275
UserMessageSignal	278
UserMessageSlot	279
ValueScaling	280
VideoWidget	280
WidgetRef	281
qcastatemachine::WriteQCaStateMachine	281
zoomMenu	282



## Chapter 9

# Class Documentation

### 9.1 \_Field Class Reference

#### Public Member Functions

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

#### Public Attributes

- [QEWidget](#) \* **qCaWidget**

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

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

## 9.2 `_Item` Class Reference

### Public Member Functions

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

### Public Attributes

- QList< [\\_Field](#) \* > **fieldList**

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

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

## 9.3 `_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

## 9.4 `_QPushButtonGroup` Class Reference

### Public Slots

- void **buttonGroupClicked** ()

### Public Member Functions

- **\_QPushButtonGroup** (QWidget \*pParent=0, QString pItemName="", QString pGroupName="", QList< [\\_Field](#) \* > \*pCurrentFieldList=0)
- void **mousePressEvent** (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

## 9.5 **\_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

## 9.6 **\_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

## 9.7 **\_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

## 9.8 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

## 9.9 QEAnalogIndicator::BandList Class Reference

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

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

## 9.10 ChartState 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**

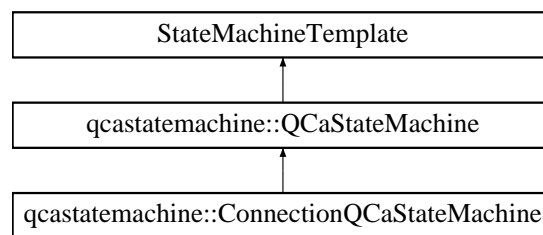
- Qt::TimeSpec **timeZoneSpec**
- QDateTime **endDateTime**

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

- /tmp/epicsqt/trunk/framework/widgets/QESTripChart/QESTripChart.cpp

## 9.11 qcastatemachine::ConnectionQCaStateMachine Class Reference

Inheritance diagram for qcastatemachine::ConnectionQCaStateMachine:



### Public Member Functions

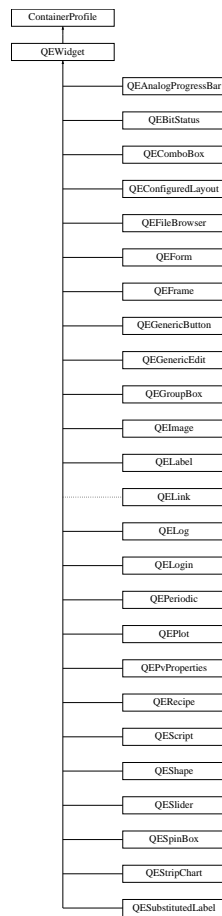
- **ConnectionQCaStateMachine** (void \*parent)
- bool **process** (int requestedState)

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

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

## 9.12 ContainerProfile Class Reference

Inheritance diagram for ContainerProfile:



## Public Member Functions

- void **takeLocalCopy** ()
- void **setupProfile** (QObject \*guiLaunchConsumerIn, QStringList pathListIn, QString parentPathIn, QString macroSubstitutionsIn)
- void **setupLocalProfile** (QObject \*guiLaunchConsumerIn, QStringList pathListIn, QString parentPathIn, QString macroSubstitutionsIn)
- void **updateConsumers** (QObject \*guiLaunchConsumerIn)
- QObject \* **replaceGuiLaunchConsumer** (QObject \*newGuiLaunchConsumerIn)
  
- void **addMacroSubstitutions** (QString macroSubstitutionsIn)
- void **removeMacroSubstitutions** ()
- 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** (QEWWidget \*containedWidget)
- QEWWidget \* **getNextContainedWidget** ()
- void **removeContainedWidget** (QEWWidget \*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)
- PersistenceManager \* **getPersistenceManager** ()

### Static Public Member Functions

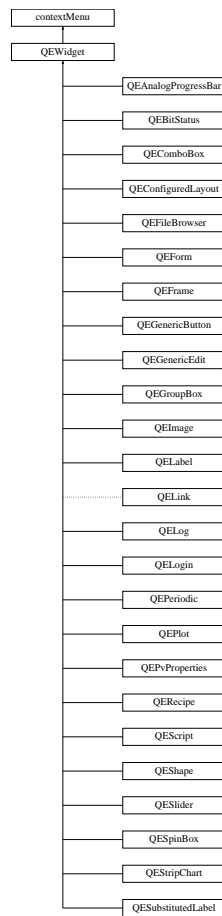
- static QChar **platformSeperator** ()

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

## 9.13 contextMenu Class Reference

Inheritance diagram for contextMenu:



## Public Types

- enum **contextMenuOptions** {  
**CM\_NONE**, **CM\_COPY\_VARIABLE**, **CM\_COPY\_DATA**, **CM\_PASTE**,  
**CM\_DRAG\_VARIABLE**, **CM\_DRAG\_DATA**, **CM\_SPECIFIC\_WIDGETS\_START\_ - HERE** }

## Public Member Functions

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



### Friends

- class [contextMenuObject](#)

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

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

## 9.14 contextMenuObject Class Reference

### Public Slots

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

### Public Member Functions

- void **addContextMenuToWidget** (QWidget \*w)
- void **manageChecked** (bool draggingVariable)
- void **setMenu** ([contextMenu](#) \*menuIn)
- bool **isDraggingVariable** ()

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

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

## 9.15 QEP periodic::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

## 9.16 flipRotateMenu Class Reference

### Public Member Functions

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

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

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

## 9.17 imageContextMenu Class Reference

### Public Types

- enum **imageContextMenuOptions** {  
**ICM\_NONE** = contextMenu::CM\_SPECIFIC\_WIDGETS\_START\_HERE, **ICM\_SAVE**,  
**ICM\_PAUSE**, **ICM\_ENABLE\_TIME**,  
**ICM\_ENABLE\_CURSOR\_PIXEL**, **ICM\_ENABLE\_CONTRAST\_REVERSAL**, **ICM\_ENABLE\_VERT**, **ICM\_ENABLE\_HOZ**,  
**ICM\_ENABLE\_AREA**, **ICM\_ENABLE\_LINE**, **ICM\_ENABLE\_TARGET**, **ICM\_DISPLAY\_BUTTON\_BAR**,  
**ICM\_DISPLAY\_BRIGHTNESS\_CONTRAST**, **ICM\_ZOOM\_SELECTED**, **ICM\_ZOOM\_FIT**, **ICM\_ZOOM\_10**,  
**ICM\_ZOOM\_25**, **ICM\_ZOOM\_50**, **ICM\_ZOOM\_75**, **ICM\_ZOOM\_100**,  
**ICM\_ZOOM\_150**, **ICM\_ZOOM\_200**, **ICM\_ZOOM\_300**, **ICM\_ZOOM\_400**,  
**ICM\_ROTATE\_NONE**, **ICM\_ROTATE\_RIGHT**, **ICM\_ROTATE\_LEFT**, **ICM\_ROTATE\_180**,  
**ICM\_FLIP\_HORIZONTAL**, **ICM\_FLIP\_VERTICAL**, **ICM\_SELECT\_PAN**, **ICM\_SELECT\_HSLICE**,  
**ICM\_SELECT\_VSLICE**, **ICM\_SELECT\_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 }

### Public Member Functions

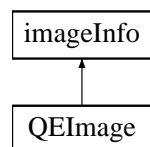
- **imageContextMenu** (QWidget \*parent=0)
- void **getContextMenuOption** (const QPoint &, imageContextMenuOptions \*option, bool \*checked)
- void **addMenuItem** (const QString &title, const bool checkable, const bool checked, const imageContextMenuOptions option)
- void **addOptionMenuItem** (const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

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

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

## 9.18 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** ()

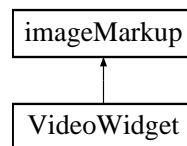
- void **infoUpdateProfile** (const QPoint start, const QPoint end, const unsigned int thickness)
- void **infoUpdateRegion** (const unsigned int region)
- void **infoUpdateRegion** (const unsigned int region, const int x1, const int y1, const int x2, const int y2)
- void **infoUpdatePixel** ()
- void **infoUpdatePixel** (const QPoint pos, int value)

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

## 9.19 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\_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)

- QCursor **getCircleCursor** ()
- QCursor **getTargetCursor** ()
- QCursor **getVLineCursor** ()
- QCursor **getHLineCursor** ()
- QCursor **getLineCursor** ()
- QCursor **getRegionCursor** ()
- virtual void **markupSetCursor** (QCursor cursor)=0

### Public Attributes

- QImage \* **markupImage**
- QVector< [markupItem](#) \* > **items**
- QPoint **grabOffset**
- bool **markupAreasStale**
- QFont **legendFont**
- QFontMetrics \* **legendFontMetrics**

### Protected Member Functions

- bool **anyVisibleMarkups** ()
- QVector< QRect > & **getMarkupAreas** ()
- QCursor **getDefaultMarkupCursor** ()
- void **setMarkupTime** ([QCaDateTime](#) &time)
- bool **markupMousePressEvent** (QMouseEvent \*event, bool panning)
- bool **markupMouseReleaseEvent** (QMouseEvent \*event, bool panning)
- bool **markupMouseMoveEvent** (QMouseEvent \*event, bool panning)
- void **markupResize** (QSize newSize, double scale)
- virtual void **markupChange** (QImage &markups, 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

## 9.20 loginWidget Class Reference

### Public Member Functions

- **loginWidget** ([QELogin](#) \*ownerIn)
- [userLevelTypes::userLevels](#) **getUserType** ()
- QString **getPassword** ()

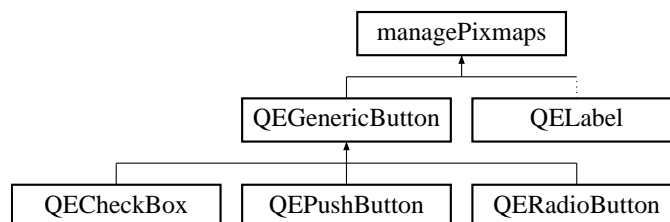
- void **clearPassword** ()

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

## 9.21 managePixmap Class Reference

Inheritance diagram for managePixmap:



### Public Member Functions

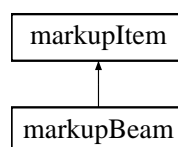
- void **setDataPixmap** (const QPixmap &Pixmap, const unsigned int index)
- QPixmap **getDataPixmap** (const unsigned int index)
- QPixmap **getDataPixmap** (const QString value)

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

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

## 9.22 markupBeam Class Reference

Inheritance diagram for markupBeam:



## Public Member Functions

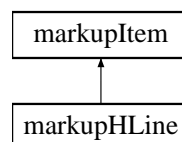
- **markupBeam** ([imageMarkup](#) \*ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor \*cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **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/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

## 9.23 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, QCursor \*cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)

- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)

### 9.23.1 Member Function Documentation

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

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

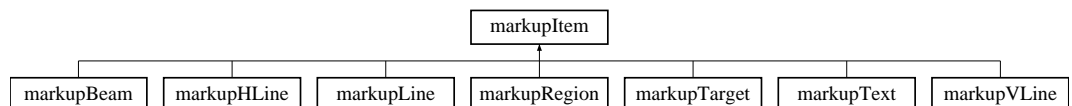
Implements [markupItem](#).

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

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

## 9.24 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 **erase** ()
- void **drawMarkupIn** ()



- void **drawMarkupOut** ()
- void **setColor** (QColor colorIn)
- void **scale** (const double xScale, const double yScale, const double zoomScale)
- 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 unsigned int **getThickness** ()=0
- virtual void **setThickness** (const unsigned int thicknessIn)=0
- virtual QCursor **defaultCursor** ()=0
- virtual void **nonInteractiveUpdate** (QRect)

### 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 QString **getLegend** ()
- 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

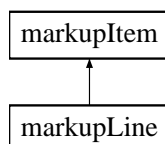
- markupHandles **activeHandle**
- isOverOptions **isOverType**
- bool **highlighted**
- int **highlightMargin**
- [imageMarkup](#) \* **owner**

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

## 9.25 markupLine Class Reference

Inheritance diagram for markupLine:



### Public Member Functions

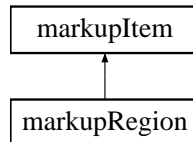
- **markupLine** ([imageMarkup](#) \*ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor \*cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **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/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

## 9.26 markupRegion Class Reference

Inheritance diagram for markupRegion:



### Public Member Functions

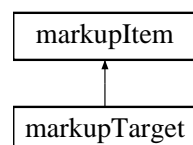
- **markupRegion** ([imageMarkup](#) \*ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point,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

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

## 9.27 markupTarget Class Reference

Inheritance diagram for markupTarget:



## Public Member Functions

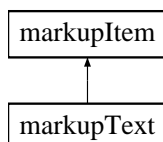
- **markupTarget** ([imageMarkup](#) \*ownerIn, const bool interactiveIn, const bool reportOnMoveIn, const QString legendIn)
- void **startDrawing** (const QPoint pos)
- void **setArea** ()
- void **drawMarkup** (QPainter &p)
- void **moveTo** (const QPoint pos)
- bool **isOver** (const QPoint point, QCursor \*cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **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/QEImage/imageMarkup.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/imageMarkup.cpp

## 9.28 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, bool draw)
- 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** ()

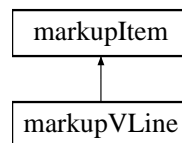
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)

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

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

## 9.29 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, QCursor \*cursor)
- QPoint **origin** ()
- QCursor **cursorForHandle** (const markupItem::markupHandles handle)
- QPoint **getPoint1** ()
- QPoint **getPoint2** ()
- unsigned int **getThickness** ()
- void **setThickness** (const unsigned int thicknessIn)
- QCursor **defaultCursor** ()
- void **scaleSpecific** (const double xScale, const double yScale, const double zoomScale)

### 9.29.1 Member Function Documentation

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

## 9.30 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

## 9.31 QEStripChartToolBar::OwnWidgets Class Reference

### Public Member Functions

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

### Public Attributes

- QPushButton \* **pushButtons** [NUMBER\_OF\_BUTTONS]
- QLabel \* **timeStatus**

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

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

## 9.32 QEPvProperties::OwnWidgets Class Reference

### Public Member Functions

- **OwnWidgets** ([QEPvProperties](#) \*parent)

### Public Attributes

- QFrame \* **topFrame**
- QLabel \* **label1**
- QLabel \* **label2**
- QLabel \* **label3**
- QLabel \* **label4**
- QLabel \* **label5**
- QLabel \* **label6**
- QComboBox \* **box**
- [QELabel](#) \* **valueLabel**
- QLabel \* **hostName**
- QLabel \* **fieldType**
- QLabel \* **timeStamp**
- QLabel \* **indexInfo**
- QVBoxLayout \* **topFrameVlayout**
- QHBoxLayout \* **hlayouts** [6]
- QWidget \* **table**
- QMenu \* **tableContextMenu**
- QFrame \* **enumerationFrame**
- [QLabelList](#) **enumerationLabelList**
- QScrollArea \* **enumerationScroll**
- QResizeableFrame \* **enumerationResize**
- QVBoxLayout \* **vlayout**

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvProperties.cpp

## 9.33 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

## 9.34 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

## 9.35 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

## 9.36 PersistenceManager 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)
- [PMElement](#) **addNamedConfiguration** (QString name)
- [PMElement](#) **getNamedConfiguration** (QString name)
- QStringList **getConfigNames** (QString fileName, QString rootName)
- void **deleteConfigs** (QString fileName, QString rootName, QStringList names)



## Friends

- class [PMElement](#)

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

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

## 9.37 PMContext Class Reference

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

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

## 9.38 PMEElement Class Reference

### Public Member Functions

- **PMEElement** ([PersistenceManager](#) \*ownerIn, QDomElement elementIn)
- **PMEElement 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)
- **PMEElement getElement** (QString name)
- **PMEElement getElement** (QString name, int i)
- **PMEElement getElement** (QString name, QString attrName, QString attrValue)
- **PMEElement getElement** (QString name, QString attrName, int attrValue)
- **PMEElementList 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/persistenceManager.h
- /tmp/epicsqt/trunk/framework/widgets/src/persistenceManager.cpp

## 9.39 PMLElementList Class Reference

### Public Member Functions

- **PMLElementList** ([PersistenceManager](#) \*ownerIn, QDomNodeList elementListIn)
- [PMElement](#) **getElement** (int i)
- int **count** ()

### 9.39.1 Member Function Documentation

#### 9.39.1.1 PMElement PMLElementList::getElement ( int i )

!! check range of i

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

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

## 9.40 QEStriptChart::PrivateData Class Reference

### Public Member Functions

- **PrivateData** ([QEStriptChart](#) \*chartIn)
- [QEStriptChartItem](#) \* **getItem** (unsigned int slot)
- QwtPlotCurve \* **allocateCurve** ()
- void **calcDisplayMinMax** ()
- void **plotData** ()
- void **setReadOut** (const QString &text)
- void **setNormalBackground** (bool state)
- void **chartContextMenuRequested** (const QPoint &pos)
- void **nullContextMenuRequested** (const QPoint &pos)
- void **chartContextMenuTriggered** (QAction \*action)
- void **itemContextMenuRequested** (const unsigned int slot, const QPoint &pos)
- void **itemContextMenuSelected** (const unsigned int, const QEStriptChartNames::ContextMenuOptions option)
- void **pushState** ()
- void **prevState** ()
- void **nextState** ()
- void **captureState** ([ChartState](#) &chartState)
- void **applyState** (const [ChartState](#) &chartState)

### Public Attributes

- QESTripChartNames::ChartYRanges **chartYScale**
- QESTripChartNames::YScaleModes **yScaleMode**
- QESTripChartNames::ChartTimeModes **chartTimeMode**
- double **timeScale**
- QString **timeUnits**

### Protected Member Functions

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

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

- /tmp/epicsqt/trunk/framework/widgets/QESTripChart/QESTripChart.cpp

## 9.41 QESTripChartItem::PrivateData Class Reference

### Public Attributes

- [QESTripChart](#) \* **chart**
- QLabel \* **pvName**
- [QELabel](#) \* **caLabel**
- QColorDialog \* **colourDialog**
- [qcaobject::QCaObject](#) \* **previousQcaltem**

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

- /tmp/epicsqt/trunk/framework/widgets/QESTripChart/QESTripChartItem.cpp

## 9.42 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/QEImage/profilePlot.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/profilePlot.cpp

## 9.43 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**
- [PersistenceManager](#) **persistenceManager**
- bool **dontActivateYet**
- bool **userLevelPasswordsSet**

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

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

## 9.44 PushButtonSpecifications Struct Reference

### Public Attributes

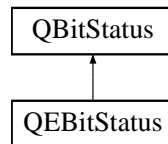
- int **gap**
- int **width**
- bool **isIcon**
- const QString **captionOrIcon**
- const QString **toolTip**
- const char \* **member**

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

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

## 9.45 QBitStatus Class Reference

Inheritance diagram for QBitStatus:



### Public Types

- enum **Orientations** { **LSB\_On\_Right**, **LSB\_On\_Bottom**, **LSB\_On\_Left**, **LSB\_On\_Top** }
- enum **Shapes** { **Rectangle**, **Circle** }

### Public Slots

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

### Public Member Functions

- **QBitStatus** (QWidget \*parent=0)
- virtual QSize **sizeHint** () const
- void **setBorderColour** (const QColor value)
- QColor **getBorderColour** ()
- void **setOnColour** (const QColor value)
- QColor **getOnColour** ()
- void **setOffColour** (const QColor value)
- QColor **getOffColour** ()
- void **setInvalidColour** (const QColor value)
- QColor **getInvalidColour** ()
- void **setClearColour** (const QColor value)
- QColor **getClearColour** ()
- void **setDrawBorder** (const bool value)
- bool **getDrawBorder** ()
- void **setNumberOfBits** (const int value)
- int **getNumberOfBits** ()
- void **setGap** (const int value)
- int **getGap** ()
- void **setShift** (const int value)
- int **getShift** ()
- void **setOnClearMask** (const QString value)
- QString **getOnClearMask** ()

- void **setOffClearMask** (const QString value)
- QString **getOffClearMask** ()
- void **setReversePolarityMask** (const QString value)
- QString **getReversePolarityMask** ()
- void **setIsValid** (const bool value)
- bool **getIsValid** ()
- void **setOrientation** (const enum Orientations value)
- enum Orientations **getOrientation** ()
- void **setShape** (const enum Shapes value)
- enum Shapes **getShape** ()
- int **getValue** ()

### Properties

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

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

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

## 9.46 QCaAlarmInfo Class Reference

### Public Member Functions

- **QCaAlarmInfo** (unsigned short statusIn, unsigned short severityIn)
- QString **statusName** ()
- QString **severityName** ()
- bool **isInAlarm** ()
- bool **isMinor** ()

- bool **isMajor** ()
- bool **isInvalid** ()
- QString **style** ()
- QString **getColorName** ()
- QCAALARMINFO\_SEVERITY **getSeverity** ()

### Static Public Member Functions

- static QCAALARMINFO\_SEVERITY **getInvalidSeverity** ()

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

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

## 9.47 QCaConnectionInfo Class Reference

### Public Member Functions

- **QCaConnectionInfo** (unsigned short channelStatIn, unsigned short linkStatIn)
- bool **isChannelConnected** ()
- bool **isLinkUp** ()

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

## 9.48 QCaDataPoint Class Reference

### Public Member Functions

- bool **isDisplayable** ()

### 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

## 9.49 QCaDataPointList Class Reference

### Public Member Functions

- void **resample** (const [QCaDataPointList](#) &source, const double interval, const [QCaDateTime](#) &endTime)

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

## 9.50 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

### 9.50.1 Member Function Documentation

#### 9.50.1.1 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

## 9.51 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

## 9.52 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

## 9.53 QCaEventUpdate Class Reference

### Public Member Functions

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

### Public Attributes

- bool **acceptThisEvent**
- [qcaobject::QCaObject](#) \* **emitterObject**
- long **reason**
- void \* **dataPtr**

### Static Public Attributes

- static QEvent::Type **EVENT\_UPDATE\_TYPE** = QEvent::User

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

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

## 9.54 QCalInstalledFiltersListItem Class Reference

### Public Member Functions

- **QCalInstalledFiltersListItem** (QObject \*eventObjectIn)

### Public Attributes

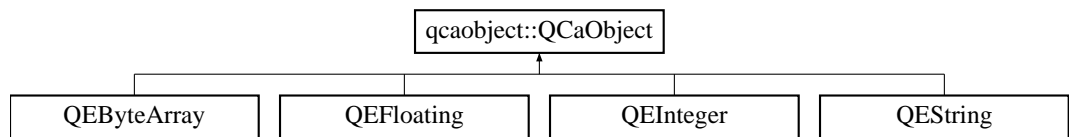
- QObject \* **eventObject**
- long **referenceCount**

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

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

## 9.55 qcaobject::QCaObject Class Reference

Inheritance diagram for qcaobject::QCaObject:



### Public Slots

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

### Signals

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

### Public Member Functions

- **QCaObject** (const QString &recordName, QObject \*eventObject, unsigned char signalsToSendIn=SIG\_VARIANT)

- **QCaObject** (const QString &recordName, QObject \*eventObject, [UserMessage](#) \*userMessageIn, unsigned char signalsToSendIn=SIG\_VARIANT)
- bool **subscribe** ()
- bool **singleShotRead** ()
- bool **dataTypeKnown** ()
- bool **createChannel** ()
- void **deleteChannel** ()
- bool **createSubscription** ()
- bool **getChannel** ()
- bool **putChannel** ()
- bool **isChannelConnected** ()
- void **startConnectionTimer** ()
- void **stopConnectionTimer** ()
- void **setUserMessage** ([UserMessage](#) \*userMessageIn)
- void **enableWriteCallbacks** (bool enable)
- bool **isWriteCallbacksEnabled** ()
- QString **getRecordName** ()
- QString **getEgu** ()
- QStringList **getEnumerations** ()
- unsigned int **getPrecision** ()
- [QCaAlarmInfo](#) **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, [QCaAlarmInfo](#) &alarmInfo, [QCaDateTime](#) &timeStamp)

### Static Public Member Functions

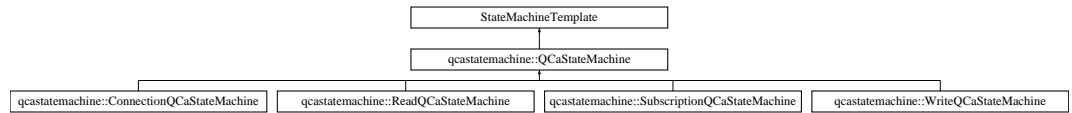
- static void **processEventStatic** ([QCaEventUpdate](#) \*dataUpdateEvent)

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

## 9.56 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

## 9.57 QCaVariableNamePropertyManager Class Reference

### Signals

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

### Public Member Functions

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

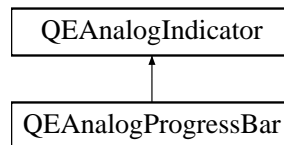
The documentation for this class was generated from the following files:

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

## 9.58 QEAnalogIndicator Class Reference

```
#include <QEAnalogIndicator.h>
```

Inheritance diagram for QEAnalogIndicator:



### Classes

- struct [Band](#)
- class [BandList](#)

### Public Types

- enum [Orientations](#) { [Left\\_To\\_Right](#), [Top\\_To\\_Bottom](#), [Right\\_To\\_Left](#), [Bottom\\_To\\_Top](#) }
- enum [Modes](#) { [Bar](#), [Scale](#), [Meter](#) }

### Public Slots

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

### Public Member Functions

- [QEAnalogIndicator](#) (QWidget \*parent=0)  
*Constructor.*
- virtual [~QEAnalogIndicator](#) ()  
*Destructor.*
- virtual QSize [sizeHint](#) () const  
*Size hint.*
- double [getValue](#) ()  
*Access function for [value](#) property - refer to [value](#) property for details.*
- void [setMinimum](#) (const double value)  
*Access function for [minimum](#) - refer to [minimum](#) property for details.*

- double [getMinimum](#) ()  
Access function for [minimum](#) - refer to [minimum](#) property for details.
- void [setMaximum](#) (const double value)  
Access function for [maximum](#) - refer to [maximum](#) property for details.
- double [getMaximum](#) ()  
Access function for [maximum](#) - refer to [maximum](#) property for details.
- void [setOrientation](#) (const enum [Orientations](#) value)  
Access function for [orientation](#) - refer to [orientation](#) property for details.
- enum [Orientations](#) [getOrientation](#) ()  
Access function for [orientation](#) - refer to [orientation](#) property for details.
- void [setMode](#) (const enum [Modes](#) value)  
Access function for [mode](#) - refer to [mode](#) property for details.
- enum [Modes](#) [getMode](#) ()  
Access function for [mode](#) - refer to [mode](#) property for details.
- void [setCentreAngle](#) (const int value)  
Access function for [centreAngle](#) - refer to [centreAngle](#) property for details.
- int [getCentreAngle](#) ()  
Access function for [centreAngle](#) - refer to [centreAngle](#) property for details.
- void [setSpanAngle](#) (const int value)  
Access function for [spanAngle](#) - refer to [spanAngle](#) property for details.
- int [getSpanAngle](#) ()  
Access function for [spanAngle](#) - refer to [spanAngle](#) property for details.
- void [setMinorInterval](#) (const double value)  
Access function for [minorInterval](#) - refer to [minorInterval](#) property for details.
- double [getMinorInterval](#) ()  
Access function for [minorInterval](#) - refer to [minorInterval](#) property for details.
- void [setMajorInterval](#) (const double value)  
Access function for [majorInterval](#) - refer to [majorInterval](#) property for details.
- double [getMajorInterval](#) ()  
Access function for [majorInterval](#) - refer to [majorInterval](#) property for details.
- void [setLogScaleInterval](#) (const int value)  
Access function for [logScaleInterval](#) - refer to [logScaleInterval](#) property for details.
- int [getLogScaleInterval](#) ()  
Access function for [logScaleInterval](#) - refer to [logScaleInterval](#) property for details.
- void [setBorderColour](#) (const QColor value)  
Access function for [borderColour](#) - refer to [borderColour](#) property for details.
- QColor [getBorderColour](#) ()  
Access function for [borderColour](#) - refer to [borderColour](#) property for details.
- void [setForegroundColour](#) (const QColor value)  
Access function for [foregroundColour](#) - refer to [foregroundColour](#) property for details.
- QColor [getForegroundColour](#) ()  
Access function for [foregroundColour](#) - refer to [foregroundColour](#) property for details.
- void [setBackgroundColour](#) (const QColor value)

- Access function for [backgroundColour](#) - refer to [backgroundColour](#) property for details.

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

  - void [setFontColour](#) (const QColor value)
- Access function for [fontColour](#) - refer to [fontColour](#) property for details.

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

  - void [setShowText](#) (const bool value)
- Access function for [showText](#) - refer to [showText](#) property for details.

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

  - void [setShowScale](#) (const bool value)
- Access function for [showScale](#) - refer to [showScale](#) property for details.

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

  - void [setLogScale](#) (const bool value)
- Access function for [logScale](#) - refer to [logScale](#) property for details.

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

### Protected Member Functions

- virtual QString [getTextImage](#) ()
- virtual [BandList](#) [getBandList](#) ()

### Properties

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

### 9.58.1 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.

### 9.58.2 Member Enumeration Documentation

#### 9.58.2.1 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)

#### 9.58.2.2 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.

### 9.58.3 Property Documentation

#### 9.58.3.1 QColor QEAnalogIndicator::backgroundColour [read, write]

Background colour

#### 9.58.3.2 QColor QEAnalogIndicator::borderColour [read, write]

Border colour

#### 9.58.3.3 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.



9.58.3.4 **QColor QEAnalogIndicator::fontColour** [read, write]

Font colour

9.58.3.5 **QColor QEAnalogIndicator::foregroundColour** [read, write]

Foreground colour

9.58.3.6 **bool QEAnalogIndicator::logScale** [read, write]

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

9.58.3.7 **int QEAnalogIndicator::logScaleInterval** [read, write]

Log scale interval.

9.58.3.8 **double QEAnalogIndicator::majorInterval** [read, write]

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

9.58.3.9 **double QEAnalogIndicator::maximum** [read, write]

Maximum indicated value.

9.58.3.10 **double QEAnalogIndicator::minimum** [read, write]

Minimum indicated value.

9.58.3.11 **double QEAnalogIndicator::minorInterval** [read, write]

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

9.58.3.12 **Modes QEAnalogIndicator::mode** [read, write]

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

9.58.3.13 **Orientations QEAnalogIndicator::orientation** [read, write]

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

### 9.58.3.14 bool QEAnalogIndicator::showScale [read, write]

If set, show the scale

### 9.58.3.15 bool QEAnalogIndicator::showText [read, write]

If set, show textual representation of value on the indicator

### 9.58.3.16 int QEAnalogIndicator::spanAngle [read, write]

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

### 9.58.3.17 double QEAnalogIndicator::value [read, write]

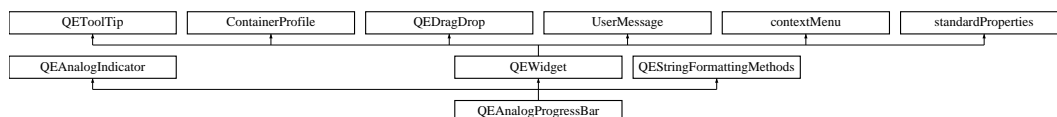
Current indicated value.

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

## 9.59 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** = 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](#) = QQStringFormatting::APPEND, [Ascii](#) = QQStringFormatting::ASCII, [Index](#) = QQStringFormatting::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 [setAlarmSeverityDisplayMode](#) (AlarmSeverityDisplayModes value)

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

- AlarmSeverityDisplayModes [getAlarmSeverityDisplayMode](#) ()

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

### Protected Member Functions

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

### Protected Attributes

- [QEFloatingFormatting](#) [floatingFormatting](#)

### Properties

- QString [variable](#)
- QString [variableSubstitutions](#)
- bool [variableAsToolTip](#)
- bool [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

### 9.59.1 Member Enumeration Documentation

#### 9.59.1.1 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.

#### 9.59.1.2 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.

#### 9.59.1.3 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.

#### 9.59.1.4 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.

### 9.59.2 Constructor & Destructor Documentation

#### 9.59.2.1 QEAnalogProgressBar::QEAnalogProgressBar ( QWidget \* *parent* = 0 )

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

#### 9.59.2.2 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.

### 9.59.3 Member Function Documentation

#### 9.59.3.1 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.

### 9.59.4 Property Documentation

#### 9.59.4.1 bool QEAnalogProgressBar::addUnits [read, write]

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

#### 9.59.4.2 AlarmSeverityDisplayModes QEAnalogProgressBar::alarmSeverityDisplayMode [read, write]

Visualise the EPICS alarm severity

**9.59.4.3** `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](#).

**9.59.4.4** `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'.

**9.59.4.5** `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](#).

**9.59.4.6** `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).

**9.59.4.7** `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.

#### 9.59.4.8 `bool QEAnalogProgressBar::leadingZero` [read, write]

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

#### 9.59.4.9 `QString QEAnalogProgressBar::localEnumeration` [read, write]

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

Format is:

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

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

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

may be included in a string to indicate a line break

Examples are:

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

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

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

#### 9.59.4.10 `Notations QEAnalogProgressBar::notation` [read, write]

Notation used for numerical formatting. Default is fixed.

#### 9.59.4.11 `int QEAnalogProgressBar::precision` [read, write]

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



**9.59.4.12** `bool QEAnalogProgressBar::trailingZeros` `[read, write]`

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

**9.59.4.13** `bool QEAnalogProgressBar::useDbDisplayLimits` `[read, write]`

Use the EPICS database display limits

**9.59.4.14** `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.

**9.59.4.15** `UserLevels QEAnalogProgressBar::userLevelEnabled` `[read, write]`

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through `setUserLevel()`. Widgets that are always 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'.

**9.59.4.16** `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.

**9.59.4.17** `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.

**9.59.4.18** `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.

#### 9.59.4.19 UserLevels QEAnalogProgressBar::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through 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'.

#### 9.59.4.20 QString QEAnalogProgressBar::variable [read, write]

EPICS variable name (CA PV)

#### 9.59.4.21 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](#).

#### 9.59.4.22 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.

#### 9.59.4.23 bool QEAnalogProgressBar::visible [read, write]

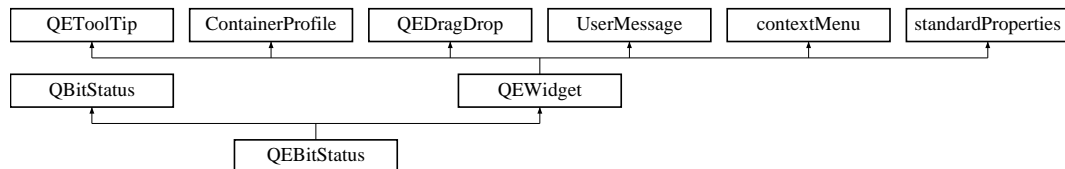
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

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

## 9.60 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](#)

## 9.60.1 Member Enumeration Documentation

### 9.60.1.1 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.

## 9.60.2 Member Function Documentation

### 9.60.2.1 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.

9.60.2.2 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](#).

### 9.60.3 Property Documentation

9.60.3.1 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](#).

9.60.3.2 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](#).

9.60.3.3 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.

9.60.3.4 UserLevels QEBitStatus::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through setUserLevel() Widgets that are always 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'.

9.60.3.5 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.

#### 9.60.3.6 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.

#### 9.60.3.7 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.

#### 9.60.3.8 UserLevels QEBitStatus::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through 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'.

#### 9.60.3.9 QString QEBitStatus::variable [read, write]

EPICS variable name (CA PV)

#### 9.60.3.10 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](#).

### 9.60.3.11 QString QEBitStatus::variableSubstitutions [read, write]

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

### 9.60.3.12 bool QEBitStatus::visible [read, write]

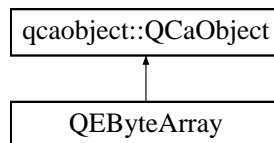
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

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

## 9.61 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

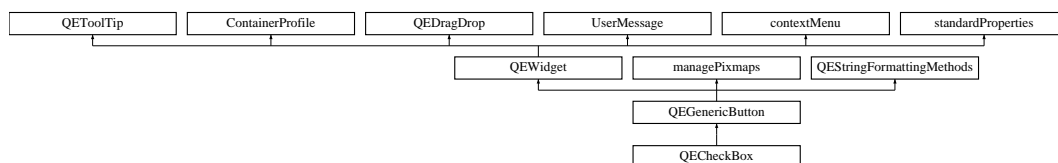
## 9.62 QEChartStateLists Class Reference

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

- /tmp/epicsqt/trunk/framework/widgets/QEstripChart/QEstripChart.cpp

## 9.63 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 `CreationOptionNames` { `Open` = `QForm::CREATION_OPTION_OPEN`, `NewTab` = `QForm::CREATION_OPTION_NEW_TAB`, `NewWindow` = `QForm::CREATION_OPTION_NEW_WINDOW` }

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

## Public Slots

- void `launchGui` (QString guiName, QForm::creationOptions creationOption)

## Signals

- void `dbValueChanged` (const QString &out)
- void `requestResend` ()  
Internal use only. Used when changing a property value to force a re-display to reflect the new property value.
- void `newGui` (QString guiName, QForm::creationOptions creationOption)  
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)

## 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](#)

- bool [writeOnPress](#)
- bool [writeOnRelease](#)
- bool [writeOnClick](#)
- QString [pressText](#)
- QString [releaseText](#)
- QString [clickText](#)
- QString [clickCheckedText](#)
- QString [labelText](#)
- QString [program](#)
- QStringList [arguments](#)
- QString [guiFile](#)
- [CreationOptionNames](#) [creationOption](#)
- QString [prioritySubstitutions](#)

### 9.63.1 Member Enumeration Documentation

#### 9.63.1.1 enum QCheckBox::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.

#### 9.63.1.2 enum QCheckBox::CreationOptionNames

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

##### Enumerator:

**Open** Replace the current GUI with the new GUI.

**NewTab** Open new GUI in a new tab.

**NewWindow** Open new GUI in a new window.

#### 9.63.1.3 enum QCheckBox::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.

#### 9.63.1.4 enum `QCheckBox::Notations`

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

##### Enumerator:

***Fixed*** Refer to [QStringFormatting::NOTATION\\_FIXED](#) for details.

***Scientific*** Refer to [QStringFormatting::NOTATION\\_SCIENTIFIC](#) for details.

***Automatic*** Refer to [QStringFormatting::NOTATION\\_AUTOMATIC](#) for details.

#### 9.63.1.5 enum `QCheckBox::UpdateOptions`

User friendly enumerations for updateOption property - refer to `QGenericButton::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)

#### 9.63.1.6 enum `QCheckBox::UserLevels`

User friendly enumerations for [userLevelVisibility](#) and [userLevelEnabled](#) properties - refer to [userLevelVisibility](#) and [userLevelEnabled](#) properties and `userLevel` enumeration for details.

##### Enumerator:

***User*** Refer to `USERLEVEL_USER` for details.

***Scientist*** Refer to `USERLEVEL_SCIENTIST` for details.

***Engineer*** Refer to `USERLEVEL_ENGINEER` for details.

### 9.63.2 Constructor & Destructor Documentation

#### 9.63.2.1 QECheckBox::QECheckBox ( QWidget \* *parent* = 0 )

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

#### 9.63.2.2 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.

### 9.63.3 Member Function Documentation

#### 9.63.3.1 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)

#### 9.63.3.2 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.

#### 9.63.3.3 void QECheckBox::launchGui ( QString *guiName*, QForm::creationOptions *creationOption* ) [inline, slot]

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

Reimplemented from [QEGenericButton](#).

#### 9.63.3.4 void QECheckBox::pressed ( int *value* ) [signal]

Button has been Pressed. The value emitted is the integer interpretation of the `press-Text` property

### 9.63.3.5 void QECheckBox::released ( int *value* ) [signal]

Button has been Released The value emitted is the integer interpretation of the release-Text property

## 9.63.4 Property Documentation

### 9.63.4.1 bool QECheckBox::addUnits [read, write]

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

### 9.63.4.2 Qt::Alignment QECheckBox::alignment [read, write]

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

### 9.63.4.3 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](#).

### 9.63.4.4 QStringList QECheckBox::arguments [read, write]

Arguments for program specified in the 'program' property.

Reimplemented from [QEGenericButton](#).

### 9.63.4.5 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'.

#### 9.63.4.6 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](#).

#### 9.63.4.7 QString QECheckBox::clickText [read, write]

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

Reimplemented from [QEGenericButton](#).

#### 9.63.4.8 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

#### 9.63.4.9 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](#).

#### 9.63.4.10 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](#).

#### 9.63.4.11 **Formats** `QCheckBox::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).

#### 9.63.4.12 **QString** `QCheckBox::guiFile` [read, write]

File name of GUI to be presented on button click. File name can be absolute, relative to the path of the QForm 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.

#### 9.63.4.13 **unsigned** `QCheckBox::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.

#### 9.63.4.14 **QString** `QCheckBox::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](#).

#### 9.63.4.15 **bool** `QCheckBox::leadingZero` [read, write]

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

#### 9.63.4.16 **QString** `QCheckBox::localEnumeration` [read, write]

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



Format is:

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

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

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

may be included in a string to indicate a line break

Examples are:

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

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

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

#### 9.63.4.17 Notations QECheckBox::notation [read, write]

Notation used for numerical formatting. Default is fixed.

#### 9.63.4.18 QString QECheckBox::password [read, write]

Password user will need to enter before any action is taken

Reimplemented from [QEGenericButton](#).

#### 9.63.4.19 QPixmap QECheckBox::pixmap0 [read, write]

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

#### 9.63.4.20 QPixmap QECheckBox::pixmap1 [read, write]

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

**9.63.4.21 QPixmap QECheckBox::pixmap2** [read, write]

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

**9.63.4.22 QPixmap QECheckBox::pixmap3** [read, write]

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

**9.63.4.23 QPixmap QECheckBox::pixmap4** [read, write]

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

**9.63.4.24 QPixmap QECheckBox::pixmap5** [read, write]

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

**9.63.4.25 QPixmap QECheckBox::pixmap6** [read, write]

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

**9.63.4.26 QPixmap QECheckBox::pixmap7** [read, write]

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

**9.63.4.27 int QECheckBox::precision** [read, write]

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

**9.63.4.28 QString QECheckBox::pressText** [read, write]

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

Reimplemented from [QEGenericButton](#).

**9.63.4.29 QString QCheckBox::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 [QGenericButton](#).

**9.63.4.30 QString QCheckBox::program** [read, write]

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

Reimplemented from [QGenericButton](#).

**9.63.4.31 QString QCheckBox::releaseText** [read, write]

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

Reimplemented from [QGenericButton](#).

**9.63.4.32 bool QCheckBox::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 [QWidget](#).

**9.63.4.33 bool QCheckBox::trailingZeros** [read, write]

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

**9.63.4.34 UpdateOptions QCheckBox::updateOption** [read, write]

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

Reimplemented from [QGenericButton](#).

**9.63.4.35 bool QCheckBox::useDbPrecision** [read, write]

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

#### 9.63.4.36 UserLevels QCheckBox::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through `setUserLevel()` Widgets that are always 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'.

#### 9.63.4.37 QString QCheckBox::userLevelEngineerStyle [read, write]

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

#### 9.63.4.38 QString QCheckBox::userLevelScientistStyle [read, write]

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

#### 9.63.4.39 QString QCheckBox::userLevelUserStyle [read, write]

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

#### 9.63.4.40 UserLevels QCheckBox::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'.

**9.63.4.41** `QString QCheckBox::variable` `[read, write]`

EPICS variable name (CA PV)

**9.63.4.42** `bool QCheckBox::variableAsToolTip` `[read, write]`

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

Reimplemented from [QEToolTip](#).

**9.63.4.43** `QString QCheckBox::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.

**9.63.4.44** `bool QCheckBox::visible` `[read, write]`

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

**9.63.4.45** `bool QCheckBox::writeOnClick` `[read, write]`

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

Reimplemented from [QEGenericButton](#).

**9.63.4.46** `bool QCheckBox::writeOnPress` `[read, write]`

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

Reimplemented from [QEGenericButton](#).

**9.63.4.47** `bool QCheckBox::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/QCheckBox.h
- /tmp/epicsqt/trunk/framework/widgets/QEButton/QCheckBox.cpp

## 9.64 QECheckBoxManager Class Reference

### Public Member Functions

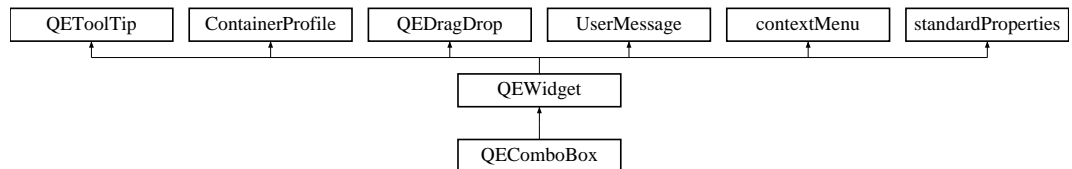
- **QECheckBoxManager** (QObject \*parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget \* **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

## 9.65 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 qlonglong &out)
- void **userChange** (const QString &oldValue, const QString &newValue, const QString &lastValue)

*Internal use only. Used by [QEConfiguredLayout](#) to be notified when one of its widgets has written something.*

## Public Member Functions

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

## 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](#)

### 9.65.1 Member Enumeration Documentation

#### 9.65.1.1 enum `QComboBox::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.

### 9.65.2 Member Function Documentation

#### 9.65.2.1 void `QComboBox::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.

### 9.65.3 Member Data Documentation

#### 9.65.3.1 bool `QComboBox::useDbEnumerations` [`read`, `write`, `protected`]

Use database enumerations - defaults to true



### 9.65.3.2 `bool QComboBox::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).

## 9.65.4 Property Documentation

### 9.65.4.1 `bool QComboBox::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](#).

### 9.65.4.2 `bool QComboBox::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](#).

### 9.65.4.3 `unsigned QComboBox::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.

### 9.65.4.4 `QString QComboBox::localEnumeration` [read, write]

Enumerations values used when `useDbEnumerations` is false.

### 9.65.4.5 `bool QComboBox::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](#).

### 9.65.4.6 `UserLevels QComboBox::userLevelEnabled` [read, write]

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

mode is set application wide through the [QELogin](#) widget, or programatically through `setUserLevel()` Widgets that are always 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'.

#### 9.65.4.7 `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.

#### 9.65.4.8 `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.

#### 9.65.4.9 `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.

#### 9.65.4.10 `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 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'.

#### 9.65.4.11 `QString QEComboBox::variable` [read, write]

EPICS variable name (CA PV)

#### 9.65.4.12 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](#).

#### 9.65.4.13 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.

#### 9.65.4.14 bool QEComboBox::visible [read, write]

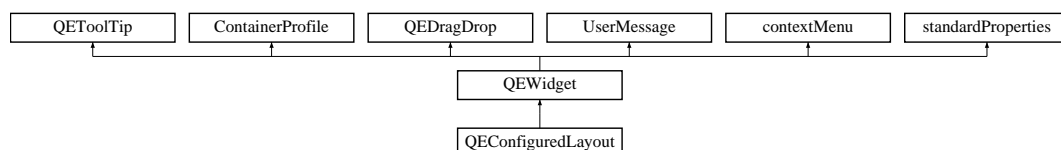
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

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

## 9.66 QEConfiguredLayout Class Reference

Inheritance diagram for QEConfiguredLayout:



### Public Types

- enum **configurationTypesProperty** { **File** = FROM\_FILE, **Text** = FROM\_TEXT }
- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **userTypesProperty** { **User** = 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 **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- void **setCurrentUserType** (int pValue)
- int **getCurrentUserType** ()
- void **refreshFields** ()
- void **userLevelChanged** ([userLevelTypes::userLevels](#) pValue)
- void **setConfigurationTypeProperty** (configurationTypesProperty pConfigurationType)
- configurationTypesProperty **getConfigurationTypeProperty** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()
- void **setCurrentUserTypeProperty** (userTypesProperty pUserType)
- userTypesProperty **getCurrentUserTypeProperty** ()

## Public Attributes

- QList< [\\_Item](#) \* > **itemList**
- QList< [\\_Field](#) \* > **currentFieldList**

## Protected Attributes

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

## Properties

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

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

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

## 9.67 QEConfiguredLayoutManager Class Reference

### Public Member Functions

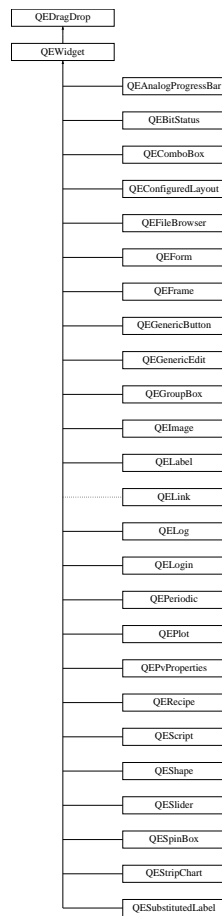
- **QEConfiguredLayoutManager** (QObject \*pParent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const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

## 9.68 QEDragDrop Class Reference

Inheritance diagram for QEDragDrop:



## Public Member Functions

- **QEDragDrop** (QWidget \*ownerIn)

## Protected Member Functions

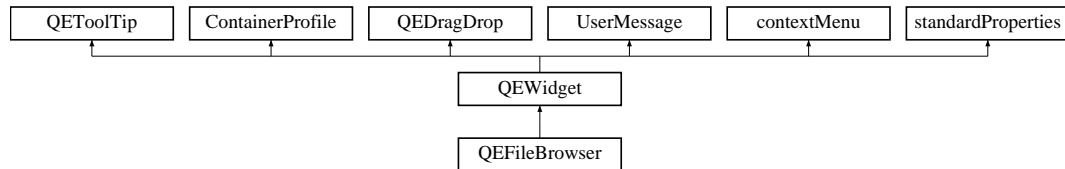
- void **qcaDragEnterEvent** (QDragEnterEvent \*event)
- void **qcaDropEvent** (QDropEvent \*event)
- void **qcaMousePressEvent** (QMouseEvent \*event)
- virtual void **setDrop** (QVariant)
- virtual QVariant **getDrop** ()
- void **setAllowDrop** (bool allowDropIn)
- bool **getAllowDrop** ()

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

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

## 9.69 QFileDialog Class Reference

Inheritance diagram for QFileDialog:



### Public Types

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

### Signals

- void **selected** (QString pFilename)

### Public Member Functions

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

### Protected Attributes

- QLineEdit \* **qlineEditDirectoryPath**
- QPushButton \* **qPushButtonDirectoryBrowser**
- QPushButton \* **qPushButtonRefresh**
- [\\_QTableWidgetFileBrowser](#) \* **qTableWidgetFileBrowser**
- QString **fileFilter**
- bool **showFileExtension**
- int **detailsLayout**

### Properties

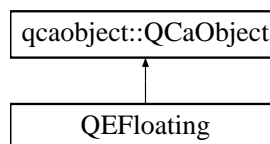
- QString **directoryPath**
- bool **showDirectoryPath**
- bool **showDirectoryBrowser**
- bool **showRefresh**
- bool **showColumnTime**
- bool **showColumnSize**
- bool **showColumnFilename**
- detailsLayoutProperty **detailsLayout**

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

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

## 9.70 QEFloating Class Reference

Inheritance diagram for QEFloating:



### Public Slots

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



## Signals

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

## Public Member Functions

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

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

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

## 9.71 QEFloatingFormatting Class Reference

### Public Types

- enum **formats** {  
**FORMAT\_e** = 'e', **FORMAT\_E** = 'E', **FORMAT\_f** = 'f', **FORMAT\_g** = 'g',  
**FORMAT\_G** = 'G' }

### Public Member Functions

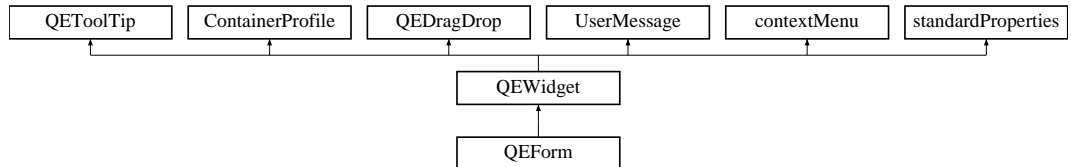
- double **formatFloating** (const QVariant &value)
- QVector< double > **formatFloatingArray** (const QVariant &value)
- QVariant **formatValue** (const double &floatingValue, generic::generic\_types valueType)
- void **setPrecision** (unsigned int precision)
- void **setFormat** (formats format)
-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

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

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

## 9.72 QForm Class Reference

Inheritance diagram for QForm:



### Public Types

- enum **creationOptions** { **CREATION\_OPTION\_OPEN**, **CREATION\_OPTION\_NEW\_TAB**, **CREATION\_OPTION\_NEW\_WINDOW** }
- enum **MessageFilterOptions** { **Match** = UserMessage::MESSAGE\_FILTER\_MATCH, **None** = UserMessage::MESSAGE\_FILTER\_NONE }

### Public Slots

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

### Public Member Functions

- **QForm** (QWidget \*parent=0)
- **QForm** (const QString &uifileNameIn, QWidget \*parent=0)
- void **commonInit** (const bool alertIfUINoFoundIn)
- QString **getQEGuiTitle** ()
- QString **getFullFileName** ()
- void **setVariableNameAndSubstitutions** (QString variableNameIn, QString variableNameSubstitutionsIn, unsigned int variableIndex)
- void **setUiFileName** (QString uiFile)
- QString **getUiFileName** ()
- void **setHandleGuiLaunchRequests** (bool handleGuiLaunchRequests)
- bool **getHandleGuiLaunchRequests** ()
- void **setResizeContents** (bool resizeContentsIn)
- bool **getResizeContents** ()
- QString **getContainedFrameworkVersion** ()
- QString **getUniqueIdentifier** ()
- void **setUniqueIdentifier** (QString name)
- void **setVariableNameSubstitutionsProperty** (QString variableNameSubstitutions)
- QString **getVariableNameSubstitutionsProperty** ()
- MessageFilterOptions **getMessageFormFilter** ()

- void **setMessageFormFilter** (MessageFilterOptions messageFormFilter)
- MessageFilterOptions **getMessageSourceFilter** ()
- void **setMessageSourceFilter** (MessageFilterOptions messageSourceFilter)

### Protected Member Functions

- void **setVariableNameSubstitutions** (QString variableNameSubstitutionsIn)

### Protected Attributes

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

### Properties

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

#### 9.72.1 Member Function Documentation

9.72.1.1 void QEForm::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.

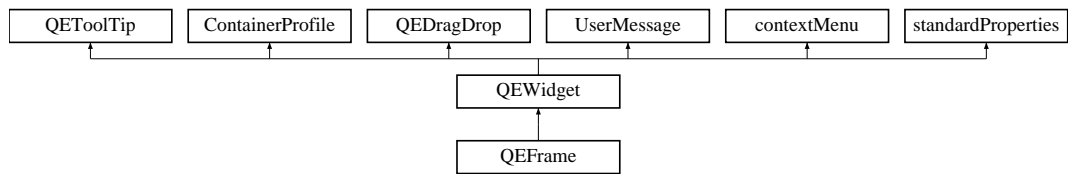
Reimplemented from [QEWidget](#).

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

## 9.73 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](#)

### 9.73.1 Member Enumeration Documentation

#### 9.73.1.1 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.

### 9.73.2 Property Documentation

#### 9.73.2.1 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](#).

#### 9.73.2.2 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](#).

#### 9.73.2.3 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.

#### 9.73.2.4 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 programatically 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'.

#### 9.73.2.5 QString QFrame::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.

#### 9.73.2.6 QString QFrame::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.

#### 9.73.2.7 QString QFrame::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.

#### 9.73.2.8 UserLevels QFrame::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'.

#### 9.73.2.9 bool QFrame::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](#).

## 9.73.2.10 bool QFrame::visible [read, write]

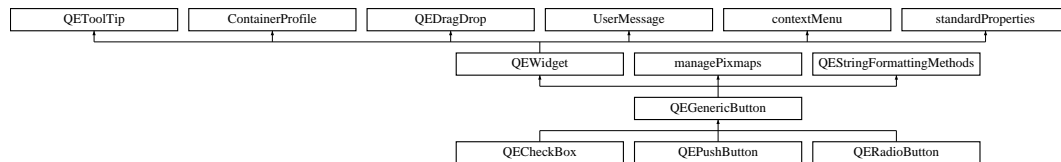
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

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

## 9.74 QEGenericButton Class Reference

Inheritance diagram for QEGenericButton:



## Public Types

- enum **updateOptions** { **UPDATE\_TEXT**, **UPDATE\_ICON**, **UPDATE\_TEXT\_AND\_ICON**, **UPDATE\_STATE** }

## Public Member Functions

- **QEGenericButton** (QWidget \*owner)
- void **setSubscribe** (bool subscribe)
- bool **getSubscribe** ()
- void **setUpdateOption** (updateOptions updateOptionIn)
- updateOptions **getUpdateOption** ()
- void **setTextAlignment** (Qt::Alignment alignment)
- Qt::Alignment **getTextAlignment** ()
- void **setPassword** (QString password)
- QString **getPassword** ()
- void **setConfirmAction** (bool confirmRequiredIn)
- bool **getConfirmAction** ()
- void **setWriteOnPress** (bool writeOnPress)
- bool **getWriteOnPress** ()
- void **setWriteOnRelease** (bool writeOnRelease)
- bool **getWriteOnRelease** ()
- void **setWriteOnClick** (bool writeOnClick)

- bool **getWriteOnClick** ()
- void **setPressText** (QString pressText)
- QString **getPressText** ()
- void **setReleaseText** (QString releaseTextIn)
- QString **getReleaseText** ()
- void **setClickText** (QString clickTextIn)
- QString **getClickText** ()
- void **setClickCheckedText** (QString clickCheckedTextIn)
- QString **getClickCheckedText** ()
- void **setProgram** (QString program)
- QString **getProgram** ()
- void **setArguments** (QStringList arguments)
- QStringList **getArguments** ()
- void **setGuiName** (QString guiName)
- QString **getGuiName** ()
- void **setPrioritySubstitutions** (QString prioritySubstitutionsIn)
- QString **getPrioritySubstitutions** ()
- void **setCreationOption** (QForm::creationOptions creationOption)
- QForm::creationOptions **getCreationOption** ()
- void **setLabelTextProperty** (QString labelTextIn)
- QString **getLabelTextProperty** ()

### Protected Member Functions

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

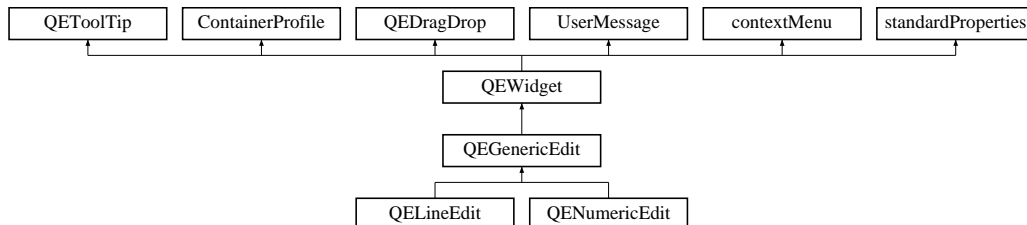
The documentation for this class was generated from the following files:

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



## 9.75 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

- [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)

### 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
- void [writeNow](#) ()

*Write the value now.*

### 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](#)

### 9.75.1 Member Enumeration Documentation

#### 9.75.1.1 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.

### 9.75.2 Constructor & Destructor Documentation

#### 9.75.2.1 QEGenericEdit::QEGenericEdit ( QWidget \* *parent* = 0 )

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

#### 9.75.2.2 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.

### 9.75.3 Member Function Documentation

#### 9.75.3.1 bool QEGenericEdit::getConfirmWrite ( )

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

#### 9.75.3.2 bool QEGenericEdit::getSubscribe ( )

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

#### 9.75.3.3 `bool QEGenericEdit::getWriteOnEnter ( )`

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

#### 9.75.3.4 `bool QEGenericEdit::getWriteOnFinish ( )`

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

#### 9.75.3.5 `bool QEGenericEdit::getWriteOnLoseFocus ( )`

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

#### 9.75.3.6 `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).

#### 9.75.3.7 `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)

#### 9.75.3.8 `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').

#### 9.75.3.9 `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).

#### 9.75.3.10 `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).

### 9.75.4 Property Documentation

**9.75.4.1 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](#).

**9.75.4.2 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).

**9.75.4.3 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](#).

**9.75.4.4 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](#).

**9.75.4.5 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](#).

**9.75.4.6 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 programatically 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'.

#### 9.75.4.7 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.

#### 9.75.4.8 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.

#### 9.75.4.9 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.

#### 9.75.4.10 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 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'.

#### 9.75.4.11 QString QEGenericEdit::variable [read, write]

EPICS variable name (CA PV)

#### 9.75.4.12 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](#).

**9.75.4.13 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.

**9.75.4.14 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.

**9.75.4.15 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').

**9.75.4.16 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).

**9.75.4.17 bool QEGenericEdit::writeOnLoseFocus** [read, write]

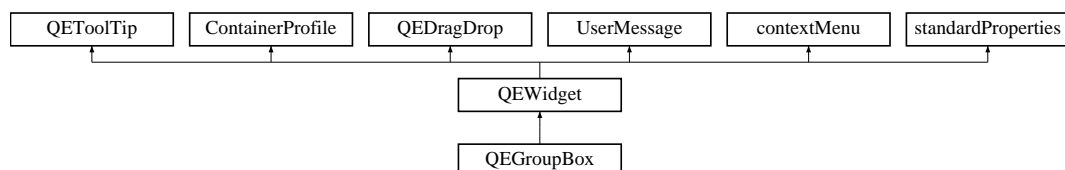
Sets if this widget automatically writes any changes when it loses focus. Default is 'false' (does not write any changes when it loses focus).

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

## 9.76 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.*
- **QEGGroupBox** (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](#)

### 9.76.1 Member Enumeration Documentation

#### 9.76.1.1 enum QEGGroupBox::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.



## 9.76.2 Property Documentation

### 9.76.2.1 `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](#).

### 9.76.2.2 `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](#).

### 9.76.2.3 `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.

### 9.76.2.4 `UserLevels QEGroupBox::userLevelEnabled` [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically 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'.

### 9.76.2.5 `QString QEGroupBox::userLevelEngineerStyle` [read, write]

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

#### 9.76.2.6 QString QEGroupBox::userLevelScientistStyle [read, write]

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

#### 9.76.2.7 QString QEGroupBox::userLevelUserStyle [read, write]

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

#### 9.76.2.8 UserLevels QEGroupBox::userLevelVisibility [read, write]

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

#### 9.76.2.9 bool QEGroupBox::variableAsToolTip [read, write]

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

Reimplemented from [QEToolTip](#).

#### 9.76.2.10 bool QEGroupBox::visible [read, write]

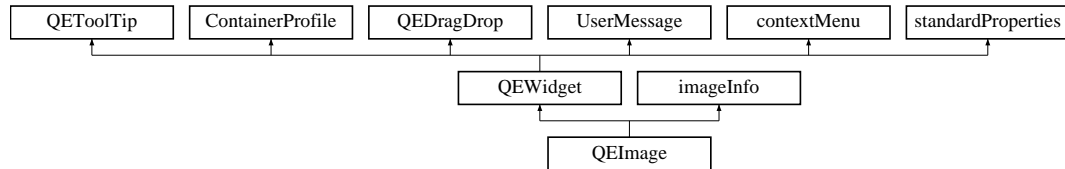
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

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

## 9.77 QImage Class Reference

Inheritance diagram for QImage:



### Classes

- struct **rgbPixel**

### 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 **formatOptions** { [GREY8](#), [GREY12](#), [GREY16](#), [RGB\\_888](#) }
- enum **resizeOptions** { [RESIZE\\_OPTION\\_ZOOM](#), [RESIZE\\_OPTION\\_FIT](#) }
- enum **rotationOptions** { [ROTATION\\_0](#), [ROTATION\\_90\\_RIGHT](#), [ROTATION\\_90\\_LEFT](#), [ROTATION\\_180](#) }
- enum **UserLevels** { [User](#) = [userLevelTypes::USERLEVEL\\_USER](#), [Scientist](#) = [userLevelTypes::USERLEVEL\\_SCIENTIST](#), [Engineer](#) = [userLevelTypes::USERLEVEL\\_ENGINEER](#) }
- enum **FormatOptions** { [Grey\\_8](#) = [QImage::GREY8](#), [Grey\\_12](#) = [QImage::GREY12](#),  
[Grey\\_16](#) = [QImage::GREY16](#), **RGB** = [QImage::RGB\\_888](#) }
- enum **ResizeOptions** { [Zoom](#) = [QImage::RESIZE\\_OPTION\\_ZOOM](#), [Fit](#) = [QImage::RESIZE\\_OPTION\\_FIT](#) }
- enum **RotationOptions** { [NoRotation](#) = [QImage::ROTATION\\_0](#), [Rotate90Right](#) = [QImage::ROTATION\\_90\\_RIGHT](#), [Rotate90Left](#) = [QImage::ROTATION\\_90\\_LEFT](#), [Rotate180](#) = [QImage::ROTATION\\_180](#) }

### Public Slots

- 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 [roi1Changed](#) ()
- Framework use only. Slot to allow external setting of selection menu options.*

  - void [roi2Changed](#) ()
- Framework use only. Slot to allow external setting of selection menu options.*

  - void [roi3Changed](#) ()
- Framework use only. Slot to allow external setting of selection menu options.*

  - void [roi4Changed](#) ()
- Framework use only. Slot to allow external setting of selection menu options.*

  - void [targetClicked](#) ()

## Signals

- void [dbValueChanged](#) (const QString &out)
  - void [requestResend](#) ()
- Internal use only. Used when changing a property value to force a re-display to reflect the new property value.*

## Public Member Functions

- [QEImage](#) (QWidget \*parent=0)
  - [QEImage](#) (const QString &variableName, QWidget \*parent=0)
  - [~QEImage](#) ()
- Destructor.*
- [selectOptions](#) [getSelectionOption](#) ()
  - void [setFormatOption](#) ([formatOptions](#) formatOption)

Access function for `#formatOption` property - refer to `#formatOption` property for details.

- [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 `initialHosScrollPos` property - refer to `initialHosScrollPos` property for details.

- int [getInitialHozScrollPos \(\)](#)

Access function for `initialHosScrollPos` property - refer to `initialHosScrollPos` 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 [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 [setDisplayCursorPixelInfo](#) (bool displayCursorPixelInfoIn)  
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 [setEnableVertSliceSelection](#) (bool enableVSliceSelectionIn)

- 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 enableHSliceSelectionIn)

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 [setEnableAreaSelection](#) (bool enableAreaSelectionIn)

Access function for [#enableAreaSelection](#) property - refer to [#enableAreaSelection](#) property for details.
- bool [getEnableAreaSelection](#) ()

Access function for [#enableAreaSelection](#) property - refer to [#enableAreaSelection](#) 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 [setEnableBrightnessContrast](#) (bool enableBrightnessContrastIn)

Access function for [enableBrightnessContrast](#) property - refer to [enableBrightnessContrast](#) property for details.
- bool [getEnableBrightnessContrast](#) ()

Access function for [enableBrightnessContrast](#) property - refer to [enableBrightnessContrast](#) 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.
- [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 [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.*

## Protected Types

- enum **variableIndexes** {  
**IMAGE\_VARIABLE, WIDTH\_VARIABLE, HEIGHT\_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, QEIMAGE\_NUM\_VARIABLES }**

## Protected Member Functions

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



- QVariant **copyData** ()
- void **paste** (QVariant v)
- void **resizeEvent** (QResizeEvent \*)

### Protected Attributes

- [QEIntegerFormatting](#) **integerFormatting**
- [resizeOptions](#) **resizeOption**
- int [zoom](#)  
*Zoom percentage. Used when #resizeOption is [Zoom](#).*
- [rotationOptions](#) **rotation**
- bool **flipVert**
- bool **flipHoz**
- int **initialHozScrollPos**
- int [initialVertScrollPos](#)
- bool [displayButtonBar](#)
- bool [enableBrightnessContrast](#)
- bool [autoBrightnessContrast](#)

### Properties

- QString [imageVariable](#)
- QString [widthVariable](#)
- QString [heightVariable](#)
- 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 [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 [showTime](#)
- QColor [vertSliceColor](#)
- QColor [hozSliceColor](#)
- QColor [profileColor](#)
- QColor [areaColor](#)
- QColor [beamColor](#)
- QColor [targetColor](#)
- QColor [timeColor](#)
- [ResizeOptions](#) [resizeOption](#)
- [RotationOptions](#) [rotation](#)
- bool [verticalFlip](#)
- bool [horizontalFlip](#)
- int [initialHosScrollPos](#)

## 9.77.1 Member Enumeration Documentation

### 9.77.1.1 enum `QEImage::formatOptions`

Video format options

#### Enumerator:

- GREY8*** 8 bit grey scale
- GREY12*** 12 bit grey scale
- GREY16*** 16 bit grey scale
- RGB\_888*** 24 bit RGB

## 9.77.1.2 enum QImage::FormatOptions

User friendly enumerations for #formatOption property - refer to #formatOption property and [formatOptions](#) enumeration for details.

**Enumerator:**

- Grey\_8** 8 bit grey scale
- Grey\_12** 12 bit grey scale
- Grey\_16** 16 bit grey scale

## 9.77.1.3 enum QImage::ResizeOptions

User friendly enumerations for #resizeOption property

**Enumerator:**

- Zoom** Zoom to selected percentage.
- Fit** Zoom to fit the current window size.

## 9.77.1.4 enum QImage::resizeOptions

Image resize options

**Enumerator:**

- RESIZE\_OPTION\_ZOOM** Zoom to selected percentage.
- RESIZE\_OPTION\_FIT** Zoom to fit the current window size.

## 9.77.1.5 enum QImage::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.

## 9.77.1.6 enum QImage::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.

**9.77.1.7 enum QImage::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.

**9.77.1.8 enum QImage::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 Constructor & Destructor Documentation****9.77.2.1 QImage::QImage ( QWidget \* parent = 0 )**

Create without a variable. Use `setVariableName'n'Property()` - where 'n' is a number from 0 to 26 - 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.

### 9.77.2.2 QImage::QImage ( 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.

## 9.77.3 Member Function Documentation

### 9.77.3.1 void QImage::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.

## 9.77.4 Member Data Documentation

### 9.77.4.1 bool QImage::autoBrightnessContrast [read, write, protected]

If true, local brightness and contrast controls are displayed. The brightness and contrast is set to use the full range of pixels in the selected area.

### 9.77.4.2 bool QImage::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.

### 9.77.4.3 bool QImage::enableBrightnessContrast [read, write, protected]

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.

### 9.77.4.4 int QImage::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.

## 9.77.5 Property Documentation

### 9.77.5.1 bool QImage::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](#).

#### 9.77.5.2 QColor QImage::areaColor [read, write]

Used to select the color of the area selection markups.

#### 9.77.5.3 QColor QImage::beamColor [read, write]

Used to select the color of the beam marker.

#### 9.77.5.4 QString QImage::beamXVariable [read, write]

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

#### 9.77.5.5 QString QImage::beamYVariable [read, write]

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

#### 9.77.5.6 QString QImage::clippingHighVariable [read, write]

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

#### 9.77.5.7 QString QImage::clippingLowVariable [read, write]

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

#### 9.77.5.8 QString QImage::clippingOnOffVariable [read, write]

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

#### 9.77.5.9 bool QImage::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](#).

**9.77.5.10** `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.

**9.77.5.11** `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 vertical pixel profile.

**9.77.5.12** `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.)

**9.77.5.13** `QString QEImage::heightVariable` [read, write]

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

**9.77.5.14** `bool QEImage::horizontalFlip` [read, write]

If true, flip image horizontally.

**9.77.5.15** `QColor QEImage::hozSliceColor` [read, write]

Used to select the color of the horizontal slice markup.

**9.77.5.16** `QString QEImage::imageVariable` [read, write]

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

**9.77.5.17** `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.

**9.77.5.18** `unsigned QEImage::int` [read, write]

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

**9.77.5.19 QColor QImage::profileColor** [read, write]

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

**9.77.5.20 QString QImage::regionOfInterest1HVariable** [read, write]

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

**9.77.5.21 QString QImage::regionOfInterest1WVariable** [read, write]

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

**9.77.5.22 QString QImage::regionOfInterest1XVariable** [read, write]

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

**9.77.5.23 QString QImage::regionOfInterest1YVariable** [read, write]

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

**9.77.5.24 QString QImage::regionOfInterest2HVariable** [read, write]

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

**9.77.5.25 QString QImage::regionOfInterest2WVariable** [read, write]

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

**9.77.5.26 QString QImage::regionOfInterest2XVariable** [read, write]

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

**9.77.5.27 QString QImage::regionOfInterest2YVariable** [read, write]

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



**9.77.5.28 QString QImage::regionOfInterest3HVariable** [read, write]

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

**9.77.5.29 QString QImage::regionOfInterest3WVariable** [read, write]

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

**9.77.5.30 QString QImage::regionOfInterest3XVariable** [read, write]

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

**9.77.5.31 QString QImage::regionOfInterest3YVariable** [read, write]

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

**9.77.5.32 QString QImage::regionOfInterest4HVariable** [read, write]

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

**9.77.5.33 QString QImage::regionOfInterest4WVariable** [read, write]

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

**9.77.5.34 QString QImage::regionOfInterest4XVariable** [read, write]

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

**9.77.5.35 QString QImage::regionOfInterest4YVariable** [read, write]

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

**9.77.5.36 ResizeOptions QImage::resizeOption** [read, write]

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

**9.77.5.37 RotationOptions QImage::rotation** [read, write]

Image rotation option.

**9.77.5.38 bool QImage::showTime** [read, write]

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

**9.77.5.39 QColor QImage::targetColor** [read, write]

Used to select the color of the target marker.

**9.77.5.40 QString QImage::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.

**9.77.5.41 QString QImage::targetXVariable** [read, write]

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

**9.77.5.42 QString QImage::targetYVariable** [read, write]

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

**9.77.5.43 QColor QImage::timeColor** [read, write]

Used to select the color of the timestamp.

**9.77.5.44 UserLevels QImage::userLevelEnabled** [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through `setUserLevel()`. Widgets that are always 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'.

**9.77.5.45 QString QImage::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.

**9.77.5.46 QString QImage::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.

**9.77.5.47 QString QImage::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.

**9.77.5.48 UserLevels QImage::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'.

**9.77.5.49 bool QImage::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](#).

#### 9.77.5.50 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.

#### 9.77.5.51 bool QEImage::verticalFlip [read, write]

If true, flip image vertically.

#### 9.77.5.52 QColor QEImage::vertSliceColor [read, write]

Used to select the color of the vertical slice markup.

#### 9.77.5.53 bool QEImage::visible [read, write]

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

#### 9.77.5.54 QString QEImage::widthVariable [read, write]

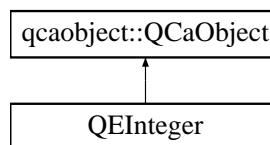
EPICS variable name (CA PV). This variable is used to read the width of the image.

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

## 9.78 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

## 9.79 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.*

### 9.79.1 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.

### 9.79.2 Member Function Documentation

#### 9.79.2.1 `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.

#### 9.79.2.2 `QVector< long > QEIntegerFormatting::formatIntegerArray ( const QVariant & value )`

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

#### 9.79.2.3 `QVariant QEIntegerFormatting::formatValue ( const long & integerValue, generic::generic_types valueType )`

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

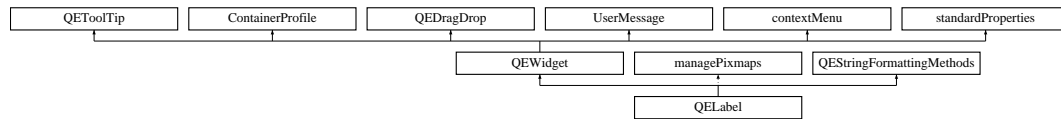
The documentation for this class was generated from the following files:

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

## 9.80 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` = `QStringFormatting::FORMAT_DEFAULT`, `Floating` = `QStringFormatting::FORMAT_FLOATING`, `Integer` = `QStringFormatting::FORMAT_INTEGER`, `UnsignedInteger` = `QStringFormatting::FORMAT_UNSIGNEDINTEGER`,  
`Time` = `QStringFormatting::FORMAT_TIME`, `LocalEnumeration` = `QStringFormatting::FORMAT_LOCAL_ENUMERATE` }
- enum `Notations` { `Fixed` = `QStringFormatting::NOTATION_FIXED`, `Scientific` = `QStringFormatting::NOTATION_SCIENTIFIC`, `Automatic` = `QStringFormatting::NOTATION_AUTOMATIC` }
- enum `ArrayActions` { `Append` = `QStringFormatting::APPEND`, `Ascii` = `QStringFormatting::ASCII`, `Index` = `QStringFormatting::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](#)

### 9.80.1 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.

### 9.80.2 Member Enumeration Documentation

#### 9.80.2.1 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.

#### 9.80.2.2 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.

#### 9.80.2.3 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.

#### 9.80.2.4 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.

#### 9.80.2.5 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.

#### 9.80.2.6 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.

### 9.80.3 Constructor & Destructor Documentation

#### 9.80.3.1 QELabel::QELabel ( QWidget \* *parent* = 0 )

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

#### 9.80.3.2 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.

## 9.80.4 Member Function Documentation

### 9.80.4.1 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.

## 9.80.5 Property Documentation

### 9.80.5.1 bool QELabel::addUnits [read, write]

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

### 9.80.5.2 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](#).

### 9.80.5.3 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'.

### 9.80.5.4 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](#).

**9.80.5.5 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).

**9.80.5.6 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.

**9.80.5.7 bool QELabel::leadingZero** [read, write]

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

**9.80.5.8 QString QELabel::localEnumeration** [read, write]

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

Format is:

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

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

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

may be included in a string to indicate a line break

Examples are:

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

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the

text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:'''

A range of numbers can be covered by a pair of values as in the following example:  $\geq 4$ :"Between 4 and 8",  $\leq 8$ :"Between 4 and 8"

#### 9.80.5.9 Notations QELabel::notation [read, write]

Notation used for numerical formatting. Default is fixed.

#### 9.80.5.10 QPixmap QELabel::pixmap0 [read, write]

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

#### 9.80.5.11 QPixmap QELabel::pixmap1 [read, write]

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

#### 9.80.5.12 QPixmap QELabel::pixmap2 [read, write]

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

#### 9.80.5.13 QPixmap QELabel::pixmap3 [read, write]

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

#### 9.80.5.14 QPixmap QELabel::pixmap4 [read, write]

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

#### 9.80.5.15 QPixmap QELabel::pixmap5 [read, write]

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

#### 9.80.5.16 QPixmap QELabel::pixmap6 [read, write]

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

**9.80.5.17 QPixmap QELabel::pixmap7** [read, write]

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

**9.80.5.18 int QELabel::precision** [read, write]

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

**9.80.5.19 bool QELabel::trailingZeros** [read, write]

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

**9.80.5.20 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.

**9.80.5.21 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.

**9.80.5.22 UserLevels QELabel::userLevelEnabled** [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through setUserLevel() Widgets that are always 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'.

**9.80.5.23 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.

**9.80.5.24 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.

**9.80.5.25 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.

**9.80.5.26 UserLevels QELabel::userLevelVisibility** [read, write]

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

**9.80.5.27 QString QELabel::variable** [read, write]

EPICS variable name (CA PV)

**9.80.5.28 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](#).

**9.80.5.29 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.



### 9.80.5.30 bool QELabel::visible [read, write]

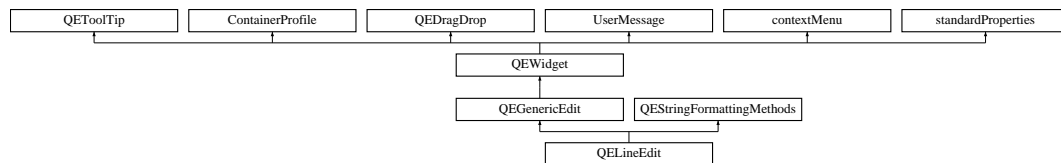
Display the widget. Default is true. Setting this property false is useful if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

## 9.81 QELineEdit Class Reference

Inheritance diagram for QELineEdit:



### Public Types

- enum [Formats](#) {  
[Default](#) = QEStrFormatting::FORMAT\_DEFAULT, [Floating](#) = QEStrFormatting::FORMAT\_FLOATING, [Integer](#) = QEStrFormatting::FORMAT\_INTEGER, [UnsignedInteger](#) = QEStrFormatting::FORMAT\_UNSIGNEDINTEGER,  
[Time](#) = QEStrFormatting::FORMAT\_TIME, [LocalEnumeration](#) = QEStrFormatting::FORMAT\_LOCAL\_ENUMERATE }
- enum [Notations](#) { [Fixed](#) = QEStrFormatting::NOTATION\_FIXED, [Scientific](#) = QEStrFormatting::NOTATION\_SCIENTIFIC, [Automatic](#) = QEStrFormatting::NOTATION\_AUTOMATIC }
- enum [ArrayActions](#) { [Append](#) = QEStrFormatting::APPEND, [Ascii](#) = QEStrFormatting::ASCII, [Index](#) = QEStrFormatting::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.
- [QLineEdit](#) (QWidget \*parent=0)
- [QLineEdit](#) (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](#)

### 9.81.1 Member Enumeration Documentation

#### 9.81.1.1 enum [QLineEdit::ArrayActions](#)

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

#### Enumerator:

***Append*** Refer to [QCStringFormatting::APPEND](#) for details.

***Ascii*** Refer to [QCStringFormatting::ASCII](#) for details.

***Index*** Refer to [QCStringFormatting::INDEX](#) for details.

## 9.81.1.2 enum QLElineEdit::Formats

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

## 9.81.1.3 enum QLElineEdit::Notations

User friendly enumerations for notation property - refer to [QQStringFormatting::notations](#) for details.

**Enumerator:**

**Fixed** Refer to [QQStringFormatting::NOTATION\\_FIXED](#) for details.

**Scientific** Refer to [QQStringFormatting::NOTATION\\_SCIENTIFIC](#) for details.

**Automatic** Refer to [QQStringFormatting::NOTATION\\_AUTOMATIC](#) for details.

## 9.81.2 Constructor &amp; Destructor Documentation

9.81.2.1 QLElineEdit::QLElineEdit ( QWidget \* *parent* = 0 )

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

9.81.2.2 QLElineEdit::QLElineEdit ( 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.

## 9.81.3 Member Function Documentation

9.81.3.1 void QLElineEdit::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.

### 9.81.4 Property Documentation

#### 9.81.4.1 `bool QLEdit::addUnits` [read, write]

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

#### 9.81.4.2 `ArrayActions QLEdit::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'.

#### 9.81.4.3 `Formats QLEdit::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).

#### 9.81.4.4 `unsigned QLEdit::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](#).

#### 9.81.4.5 `bool QLEdit::leadingZero` [read, write]

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

#### 9.81.4.6 `QString QLEdit::localEnumeration` [read, write]

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

Format is:

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

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

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

may be included in a string to indicate a line break

Examples are:

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

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

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

#### 9.81.4.7 Notations QELineEdit::notation [read, write]

Notation used for numerical formatting. Default is fixed.

#### 9.81.4.8 int QELineEdit::precision [read, write]

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

#### 9.81.4.9 bool QELineEdit::trailingZeros [read, write]

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

#### 9.81.4.10 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/QLineEdit/QLineEdit.cpp

## 9.82 QLEditManager Class Reference

### Public Member Functions

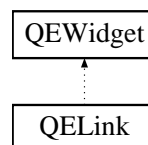
- **QLEditManager** (QObject \*parent=0)
- bool **isContainer** () const
- bool **isInitialized** () const
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget \*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

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

- /tmp/epicsqt/trunk/framework/widgets/QLineEdit/QLEditManager.h

## 9.83 QELink Class Reference

Inheritance diagram for QELink:



### Public Types

- enum **conditions** {  
**CONDITION\_EQ**, **CONDITION\_NE**, **CONDITION\_GT**, **CONDITION\_GE**,  
**CONDITION\_LT**, **CONDITION\_LE** }
- enum **ConditionNames** {  
**Equal** = QELink::CONDITION\_EQ, **NotEqual** = QELink::CONDITION\_NE, **GreaterThan**  
= QELink::CONDITION\_GT, **GreaterThanOrEqual** = QELink::CONDITION\_GE,  
**LessThan** = QELink::CONDITION\_LT, **LessThanOrEqual** = QELink::CONDITION\_-  
LE }

### Public Slots

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

### Signals

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

### Public Member Functions

- **QELink** (QWidget \*parent=0)
- void **setCondition** (conditions conditionIn)
- conditions **getCondition** ()
- void **setComparisonValue** (QString comparisonValue)
- QString **getComparisonValue** ()
- void **setSignalTrue** (bool signalTrue)
- bool **getSignalTrue** ()
- void **setSignalFalse** (bool signalFalse)
- bool **getSignalFalse** ()
- void **setOutTrueValue** (QString outTrueValue)
- QString **getOutTrueValue** ()
- void **setOutFalseValue** (QString outFalseValue)
- QString **getOutFalseValue** ()
- void **setRunVisible** (bool visibleIn)
- bool **getRunVisible** ()
- void **setConditionProperty** (ConditionNames condition)
- ConditionNames **getConditionProperty** ()

### Protected Attributes

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

## Properties

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

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

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

## 9.84 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)

### 9.84.1 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](#).



## 9.84.2 Constructor & Destructor Documentation

### 9.84.2.1 QELocalEnumeration::QELocalEnumeration ( )

Constructors

### 9.84.2.2 QELocalEnumeration::QELocalEnumeration ( const QString & *localEnumeration* )

Constructor with localEnumeration

## 9.84.3 Member Function Documentation

### 9.84.3.1 QString QELocalEnumeration::getLocalEnumeration ( )

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

### 9.84.3.2 bool QELocalEnumeration::isDefined ( )

Evaluates: `getLocalEnumeration.count() > 0`

### 9.84.3.3 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)  
>= Greather than or equal > Greater than Always match (used to specify default text)

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

may be included in a string to indicate a line break

Examples are:

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

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the

text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:'"

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

Will completely re-initialises the object.

#### 9.84.3.4 double QELocalEnumeration::textToDouble ( 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.

#### 9.84.3.5 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.

#### 9.84.3.6 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

#### 9.84.3.7 QString QELocalEnumeration::valueToText ( const QVariant & *value*, bool & *match* )

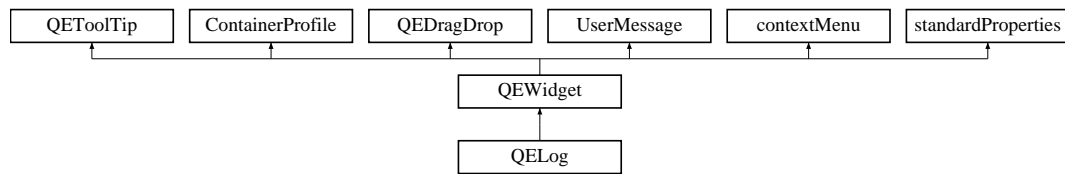
Format a variant value using local enumeration list. If the value is numeric, then the value is compared to the numeric interpretation of the enumeration values, if the value is textual, then the value is compared to the textual enumeration values.

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

## 9.85 QELog Class Reference

Inheritance diagram for QELog:



## Public Types

- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **MessageFilterOptions** { **Any** = UserMessage::MESSAGE\_FILTER\_ANY, **Match** = UserMessage::MESSAGE\_FILTER\_MATCH, **None** = UserMessage::MESSAGE\_FILTER\_NONE }

## Public Member Functions

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

- void **setMessageFormFilter** (MessageFilterOptions messageFormFilter)
- MessageFilterOptions **getMessageSourceFilter** ()
- void **setMessageSourceFilter** (MessageFilterOptions messageSourceFilter)

### Protected Attributes

- [\\_QTableWidgetLog](#) \* **qTableWidgetLog**
- QCheckBox \* **qCheckBoxInfoMessage**
- QCheckBox \* **qCheckBoxWarningMessage**
- QCheckBox \* **qCheckBoxErrorMessage**
- QPushButton \* **qPushButtonClear**
- QPushButton \* **qPushButtonSave**
- QColor **qColorInfo**
- QColor **qColorWarning**
- QColor **qColorError**
- bool **scrollToBottom**
- int **detailsLayout**

### Properties

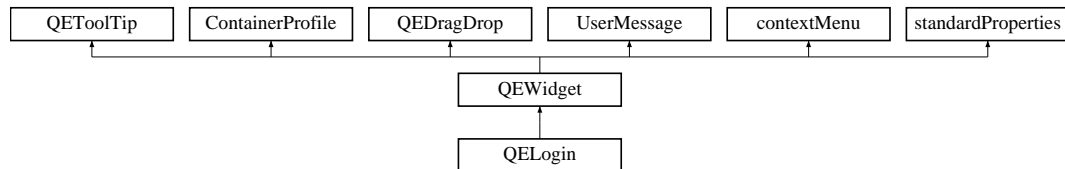
- bool **showColumnTime**
- bool **showColumnType**
- bool **showColumnMessage**
- bool **showMessageFilter**
- bool **showClear**
- bool **showSave**
- detailsLayoutProperty **detailsLayout**
- QColor **infoColor**
- QColor **warningColor**
- QColor **errorColor**
- MessageFilterOptions **messageFormFilter**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

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

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

## 9.86 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

## 9.87 QELoginDialog Class Reference

### Public Member Functions

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

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

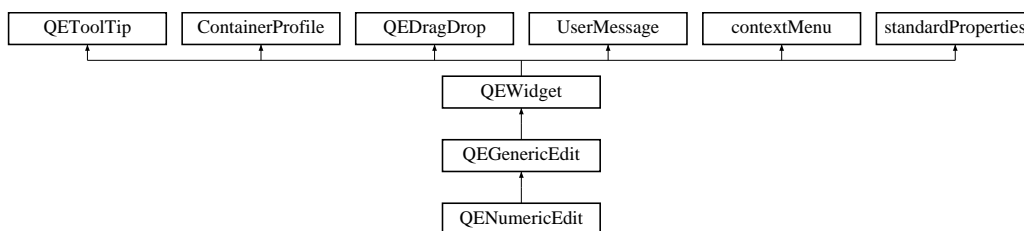
- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.h
- /tmp/epicsqt/trunk/framework/widgets/QELogin/QELogin.cpp

## 9.88 QENumericEdit Class Reference

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.

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

Inheritance diagram for QENumericEdit:



### Public Types

- enum [Radices](#) { **Decimal** = 0, **Hexadecimal**, **Octal**, **Binary** }  
*Specify radix, default is Decimal.*
- enum [Separators](#) { **None** = 0, **Comma**, **Underscore**, **Space** }  
*Specify digit 'thousands' separator character, default is none.*

### 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](#) ()

### Protected Member Functions

- 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 [Radices](#) value)
- [Radices](#) **getRadix** ()
- void **setSeparator** (const [Separators](#) value)
- [Separators](#) **getSeparator** ()
- void **keyPressEvent** (QKeyEvent \*event)
- void **focusInEvent** (QFocusEvent \*event)
- void **mouseReleaseEvent** (QMouseEvent \*event)
- void **establishConnection** (unsigned int variableIndex)
- [qcaobject::QCaObject](#) \* **createQcaltem** (unsigned int variableIndex)
- int **getPrecision** ()
- int **getLeadingZeros** ()
- double **getMinimum** ()
- double **getMaximum** ()
- int **maximumSignificance** ()
- int **getRadixValue** ()
- void **setValue** (const QVariant &value)  
*Sets the undelying QLineEdit widget to the given value.*
- QVariant **getValue** ()  
*Gets the undelying value.*
- bool **writeData** (const QVariant &value, QString &message)  
*Write the data to the channel.*

### Protected Attributes

- [QEFloatingFormatting](#) **floatingFormatting**

## Properties

- bool [autoScale](#)
- int [precision](#)
- int [leadingZeros](#)
- double [minimum](#)
- double [maximum](#)
- bool [addUnits](#)
- [Radices](#) **radix**
- [Separators](#) **separator**

## Friends

- class **NumericValidator**

### 9.88.1 Detailed Description

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

Note: this class based on thumb\_wheel\_edits.pas by same author.

### 9.88.2 Constructor & Destructor Documentation

#### 9.88.2.1 [QENumericEdit::QENumericEdit \( QWidget \\* \*parent\* = 0 \)](#)

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

#### 9.88.2.2 [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.

### 9.88.3 Member Function Documentation

#### 9.88.3.1 [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.



### 9.88.4 Property Documentation

**9.88.4.1** `bool QENumericEdit::addUnits` [read, write]

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

**9.88.4.2** `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.

**9.88.4.3** `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.

**9.88.4.4** `double QENumericEdit::maximum` [read, write]

Specifies the maximum allowed value. This is only used if autoScale is false.

**9.88.4.5** `double QENumericEdit::minimum` [read, write]

Specifies the minimum allowed value. This is only used if autoScale is false.

**9.88.4.6** `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

## 9.89 QENumericEditManager Class Reference

### Public Member Functions

- **QENumericEditManager** (QObject \*parent=0)
- `bool isContainer () const`
- `bool isInitialized () const`

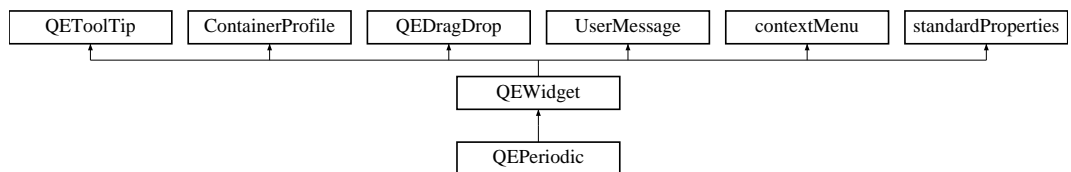
- QIcon **icon** () const
- QString **group** () const
- QString **includeFile** () const
- QString **name** () const
- QString **toolTip** () const
- QString **whatsThis** () const
- QWidget \* **createWidget** (QWidget \*parent)
- void **initialize** (QDesignerFormEditorInterface \*core)

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

- /tmp/epicsqt/trunk/framework/widgets/QLineEdit/QENumericEditManager.h
- /tmp/epicsqt/trunk/framework/widgets/QLineEdit/QENumericEditManager.cpp

## 9.90 QEPeiodic Class Reference

Inheritance diagram for QEPeiodic:



### 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 **setDrop** (QVariant drop)
- QVariant **getDrop** ()

### Protected Attributes

- [QEFloatingFormatting](#) **floatingFormatting**
- bool **localEnabled**
- bool [allowDrop](#)
- variableTypes **variableType1**
- variableTypes **variableType2**
- double **variableTolerance1**
- double **variableTolerance2**

## Properties

- QString [writeButtonVariable1](#)
- QString [writeButtonVariable2](#)
- QString [readbackLabelVariable1](#)
- QString [readbackLabelVariable2](#)
- QString [variableSubstitutions](#)
- bool [subscribe](#)
- bool [variableAsToolTip](#)
- bool [visible](#)
- unsigned [int](#)
- QString [userLevelUserStyle](#)
- QString [userLevelScientistStyle](#)
- QString [userLevelEngineerStyle](#)
- [UserLevels](#) [userLevelVisibility](#)
- [UserLevels](#) [userLevelEnabled](#)
- bool [displayAlarmState](#)
- PresentationOptions **presentationOption**
- VariableTypes **variableType1**
- VariableTypes **variableType2**
- QString **userInfo**

### 9.90.1 Member Enumeration Documentation

#### 9.90.1.1 enum QEPeiodic::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 Member Function Documentation

#### 9.90.2.1 void QEPeiodic::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.

### 9.90.2.2 void QEPeiodic::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.

## 9.90.3 Member Data Documentation

### 9.90.3.1 bool QEPeiodic::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](#).

## 9.90.4 Property Documentation

### 9.90.4.1 bool QEPeiodic::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](#).

### 9.90.4.2 unsigned QEPeiodic::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.

### 9.90.4.3 QString QEPeiodic::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.

### 9.90.4.4 QString QEPeiodic::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.

**9.90.4.5 bool QEPeiodic::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](#).

**9.90.4.6 UserLevels QEPeiodic::userLevelEnabled** [read, write]

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

**9.90.4.7 QString QEPeiodic::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.

**9.90.4.8 QString QEPeiodic::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.

**9.90.4.9 QString QEPeiodic::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.

#### 9.90.4.10 UserLevels QEPeiodic::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'.

#### 9.90.4.11 bool QEPeiodic::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](#).

#### 9.90.4.12 QString QEPeiodic::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.

#### 9.90.4.13 bool QEPeiodic::visible [read, write]

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

#### 9.90.4.14 QString QEPeiodic::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.

#### 9.90.4.15 QString QEPeiodic::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/QEPeiodic/QEPeiodic.h
- /tmp/epicsqt/trunk/framework/widgets/QEPeiodic/QEPeiodic.cpp

## 9.91 QEPeiodicComponentData 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

## 9.92 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

## 9.93 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

## 9.94 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 **valueInitd** (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

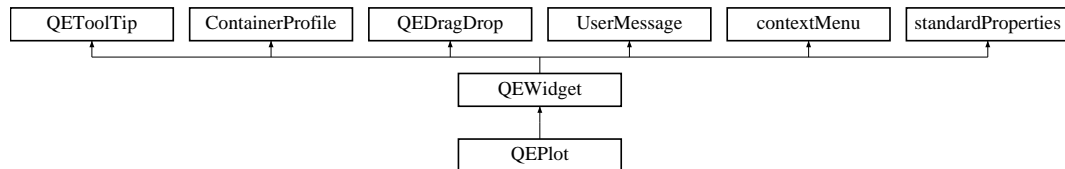
- static void **setDebugLevel** (unsigned level=0)
- static QVariant **get** (const QString &\_pvName, int delay=defaultDelay)
- static QVariant **set** (QString &\_pvName, const QVariant &value, int delay=-1)

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

## 9.95 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 [userLevel/Visibility](#) property - refer to [userLevel/Visibility](#) property for details.*
- void [setUserLevelVisibilityProperty](#) ([UserLevels](#) level)
 

*Access function for [userLevel/Visibility](#) property - refer to [userLevel/Visibility](#) property for details.*

- [UserLevels](#) [getUserLevelEnabledProperty](#) ()  
Access function for [userLevelEnabled](#) property - refer to [userLevelEnabled](#) property for details.
- void [setUserLevelEnabledProperty](#) ([UserLevels](#) level)  
Access function for [userLevelEnabled](#) property - refer to [userLevelEnabled](#) property for details.
- void **setTraceStyle1** ([TraceStyles](#) traceStyle)
- void **setTraceStyle2** ([TraceStyles](#) traceStyle)
- void **setTraceStyle3** ([TraceStyles](#) traceStyle)
- void **setTraceStyle4** ([TraceStyles](#) traceStyle)
- [TraceStyles](#) **getTraceStyle1** ()
- [TraceStyles](#) **getTraceStyle2** ()
- [TraceStyles](#) **getTraceStyle3** ()
- [TraceStyles](#) **getTraceStyle4** ()

### Protected Member Functions

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

### Protected Attributes

- [QEFloatingFormatting](#) **floatingFormatting**
- bool **localEnabled**
- bool [allowDrop](#)

### Properties

- [QString](#) [variable1](#)
- [QString](#) [variable2](#)
- [QString](#) [variable3](#)
- [QString](#) [variable4](#)
- [QString](#) [variableSubstitutions](#)
- bool [variableAsToolTip](#)
- bool [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**

### 9.95.1 Member Enumeration Documentation

#### 9.95.1.1 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.

### 9.95.2 Member Function Documentation

#### 9.95.2.1 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.

#### 9.95.2.2 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.

### 9.95.3 Member Data Documentation

#### 9.95.3.1 `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](#).

### 9.95.4 Property Documentation

#### 9.95.4.1 `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](#).

#### 9.95.4.2 `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.

#### 9.95.4.3 `UserLevels QEPlot::userLevelEnabled` [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through `setUserLevel()` Widgets that are always 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'.

#### 9.95.4.4 `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.

#### 9.95.4.5 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.

#### 9.95.4.6 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.

#### 9.95.4.7 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'.

#### 9.95.4.8 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.

#### 9.95.4.9 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.

#### 9.95.4.10 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.



## 9.95.4.11 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.

## 9.95.4.12 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](#).

## 9.95.4.13 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.

## 9.95.4.14 bool QEPlot::visible [read, write]

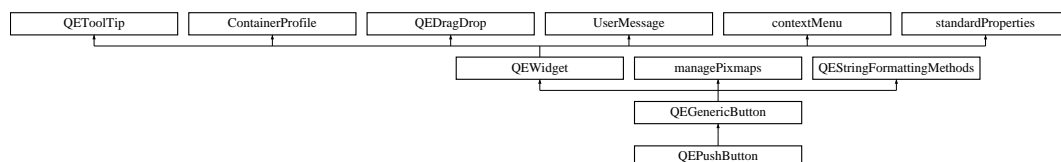
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

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

## 9.96 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](#) = QQStringFormatting::FORMAT\_DEFAULT, [Floating](#) = QQStringFormatting::FORMAT\_FLOATING, [Integer](#) = QQStringFormatting::FORMAT\_INTEGER, [UnsignedInteger](#) = QQStringFormatting::FORMAT\_UNSIGNEDINTEGER,  
[Time](#) = QQStringFormatting::FORMAT\_TIME, [LocalEnumeration](#) = QQStringFormatting::FORMAT\_LOCAL\_ENUMERATE }
- enum [Notations](#) { [Fixed](#) = QQStringFormatting::NOTATION\_FIXED, [Scientific](#) = QQStringFormatting::NOTATION\_SCIENTIFIC, [Automatic](#) = QQStringFormatting::NOTATION\_AUTOMATIC }
- enum [ArrayActions](#) { [Append](#) = QQStringFormatting::APPEND, [Ascii](#) = QQStringFormatting::ASCII, [Index](#) = QQStringFormatting::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 [CreationOptionNames](#) { [Open](#) = QForm::CREATION\_OPTION\_OPEN, [NewTab](#) = QForm::CREATION\_OPTION\_NEW\_TAB, [NewWindow](#) = QForm::CREATION\_OPTION\_NEW\_WINDOW }  
*Creation options. Used to indicate how to present a GUI when requesting a new GUI be created. Open a new window, open a new tab, or replace the current window.*

## Public Slots

- void [launchGui](#) (QString guiName, QForm::creationOptions creationOption)

## Signals

- void [dbValueChanged](#) (const QString &out)
- void [requestResend](#) ()  
*Internal use only. Used when changing a property value to force a re-display to reflect the new property value.*
- void [newGui](#) (QString guiName, QForm::creationOptions creationOption)  
*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)

## 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](#)
- bool [writeOnPress](#)
- bool [writeOnRelease](#)
- bool [writeOnClick](#)
- QString [pressText](#)
- QString [releaseText](#)
- QString [clickText](#)
- QString [clickCheckedText](#)
- QString [labelText](#)
- QString [program](#)
- QStringList [arguments](#)
- QString [guiFile](#)
- [CreationOptionNames](#) creationOption
- QString [prioritySubstitutions](#)

### 9.96.1 Member Enumeration Documentation

#### 9.96.1.1 enum [QEPushButton::ArrayActions](#)

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

##### Enumerator:

***Append*** Refer to [QCStringFormatting::APPEND](#) for details.

***Ascii*** Refer to [QCStringFormatting::ASCII](#) for details.

***Index*** Refer to [QCStringFormatting::INDEX](#) for details.

## 9.96.1.2 enum QEPushButton::CreationOptionNames

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

**Enumerator:**

**Open** Replace the current GUI with the new GUI.

**NewTab** Open new GUI in a new tab.

**NewWindow** Open new GUI in a new window.

## 9.96.1.3 enum QEPushButton::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.

## 9.96.1.4 enum QEPushButton::Notations

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

**Enumerator:**

**Fixed** Refer to [QStringFormatting::NOTATION\\_FIXED](#) for details.

**Scientific** Refer to [QStringFormatting::NOTATION\\_SCIENTIFIC](#) for details.

**Automatic** Refer to [QStringFormatting::NOTATION\\_AUTOMATIC](#) for details.

## 9.96.1.5 enum QEPushButton::UpdateOptions

User friendly enumerations for updateOption property - refer to [QEGenericButton::updateOptions](#) for details.

**Enumerator:**

**Text** Data updates will update the button text.

**Icon** Data updates will update the button icon.

**TextAndIcon** Data updates will update the button text and icon.

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

### 9.96.1.6 enum QEPushButton::UserLevels

User friendly enumerations for [userLevelVisibility](#) and [userLevelEnabled](#) properties - refer to [userLevelVisibility](#) and [userLevelEnabled](#) properties and #userLevel enumeration for details.

#### Enumerator:

**User** Refer to USERLEVEL\_USER for details.

**Scientist** Refer to USERLEVEL\_SCIENTIST for details.

**Engineer** Refer to USERLEVEL\_ENGINEER for details.

## 9.96.2 Constructor & Destructor Documentation

### 9.96.2.1 QEPushButton::QEPushButton ( QWidget \* *parent* = 0 )

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

### 9.96.2.2 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.

## 9.96.3 Member Function Documentation

### 9.96.3.1 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)

### 9.96.3.2 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.

### 9.96.3.3 void QEPushButton::launchGui ( QString *guiName*, QForm::creationOptions *creationOption* ) [inline, slot]

Default slot used to create a new GUI if there is no slot indicated in the [ContainerProfile](#) class. This slot is typically used when the button is pressed within the Designer preview window to allow the operation of the button to be tested. If an application does not

specify a slot to use for creating new windows (through the [ContainerProfile](#) class) a window will still be created through this slot, but it will not respect the window creation options or any other window related application constraints. For example, the QEGui application does provide a slot for creating new GUIs in the [ContainerProfile](#) class which respects the creation options, knows how to add tabs in the application, and extend the application's window menu in the menu bar.

Reimplemented from [QEGenericButton](#).

**9.96.3.4** `void QEPushButton::pressed ( int value ) [signal]`

Button has been Pressed. The value emitted is the integer interpretation of the press-Text property

**9.96.3.5** `void QEPushButton::released ( int value ) [signal]`

Button has been Released The value emitted is the integer interpretation of the release-Text property

## 9.96.4 Property Documentation

**9.96.4.1** `bool QEPushButton::addUnits [read, write]`

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

**9.96.4.2** `Qt::Alignment QEPushButton::alignment [read, write]`

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

**9.96.4.3** `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](#).

**9.96.4.4** `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.

**9.96.4.5** `QStringList QEPushButton::arguments [read, write]`

Arguments for program specified in the 'program' property.

Reimplemented from [QEGenericButton](#).

#### 9.96.4.6 **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'.

#### 9.96.4.7 **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](#).

#### 9.96.4.8 **QString QEPushButton::clickText** [read, write]

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

Reimplemented from [QEGenericButton](#).

#### 9.96.4.9 **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



**9.96.4.10 CreationOptionNames QEPushButton::creationOption** [read, write]

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

Reimplemented from [QEGenericButton](#).

**9.96.4.11 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](#).

**9.96.4.12 Formats QEPushButton::format** [read, write]

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

**9.96.4.13 QString QEPushButton::guiFile** [read, write]

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

**9.96.4.14 unsigned QEPushButton::int** [read, write]

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

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

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

**9.96.4.15 QString QEPushButton::labelText** [read, write]

Button label text (prior to substitution). Macro substitutions will be applied to this text and the result will be set as the button text. Used when data updates are not being represented in the button text. 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](#).

**9.96.4.16 bool QEPushButton::leadingZero** [read, write]

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

**9.96.4.17 QString QEPushButton::localEnumeration** [read, write]

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

Format is:

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

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

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

may be included in a string to indicate a line break

Examples are:

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

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

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

**9.96.4.18 Notations QEPushButton::notation** [read, write]

Notation used for numerical formatting. Default is fixed.

**9.96.4.19 QString QEPushButton::password** [read, write]

Password user will need to enter before any action is taken

Reimplemented from [QEGenericButton](#).

**9.96.4.20 QPixmap QEPushButton::pixmap0** [read, write]

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

**9.96.4.21 QPixmap QEPushButton::pixmap1** [read, write]

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

**9.96.4.22 QPixmap QEPushButton::pixmap2** [read, write]

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

**9.96.4.23 QPixmap QEPushButton::pixmap3** [read, write]

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

**9.96.4.24 QPixmap QEPushButton::pixmap4** [read, write]

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

**9.96.4.25 QPixmap QEPushButton::pixmap5** [read, write]

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

**9.96.4.26 QPixmap QEPushButton::pixmap6** [read, write]

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

#### 9.96.4.27 QPixmap QEPushButton::pixmap7 [read, write]

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

#### 9.96.4.28 int QEPushButton::precision [read, write]

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

#### 9.96.4.29 QString QEPushButton::pressText [read, write]

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

Reimplemented from [QEGenericButton](#).

#### 9.96.4.30 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 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 substittions applied to the new form, they are appended to the list and the existing macro substitutions take precedence.

Reimplemented from [QEGenericButton](#).

#### 9.96.4.31 QString QEPushButton::program [read, write]

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

Reimplemented from [QEGenericButton](#).

#### 9.96.4.32 QString QEPushButton::releaseText [read, write]

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

Reimplemented from [QEGenericButton](#).

#### 9.96.4.33 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](#).

**9.96.4.34** `bool QPushButton::trailingZeros` [read, write]

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

**9.96.4.35** `UpdateOptions QPushButton::updateOption` [read, write]

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

Reimplemented from [QEGenericButton](#).

**9.96.4.36** `bool QPushButton::useDbPrecision` [read, write]

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

**9.96.4.37** `UserLevels QPushButton::userLevelEnabled` [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through `setUserLevel()` Widgets that are always 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'.

**9.96.4.38** `QString QPushButton::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.

**9.96.4.39** `QString QPushButton::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.

#### 9.96.4.40 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.

#### 9.96.4.41 UserLevels QEPushButton::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through 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'.

#### 9.96.4.42 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.

#### 9.96.4.43 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](#).

#### 9.96.4.44 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.

#### 9.96.4.45 bool QEPushButton::visible [read, write]

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

#### 9.96.4.46 bool QEPushButton::writeOnClick [read, write]

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

Reimplemented from [QEGenericButton](#).

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

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

Reimplemented from [QEGenericButton](#).

9.96.4.48 `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

## 9.97 QEPVNameLists Class Reference

### Public Member Functions

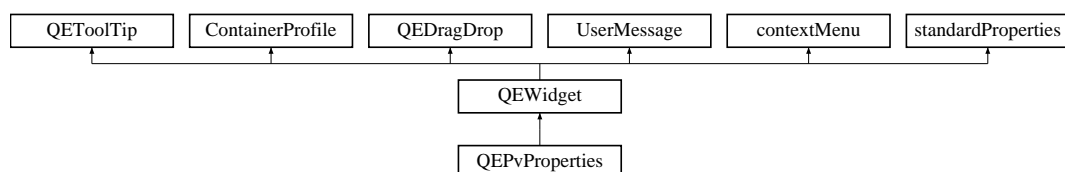
- void **prependOrMoveToFirst** (const QString &item)
- void **saveConfiguration** ([PMElement](#) &parentElement)
- void **restoreConfiguration** ([PMElement](#) &parentElement)

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

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

## 9.98 QEPvProperties Class Reference

Inheritance diagram for QEPvProperties:



### Classes

- class [OwnWidgets](#)

## Public Types

- enum [UserLevels](#) { [User](#) = userLevelTypes::USERLEVEL\_USER, [Scientist](#) = userLevelTypes::USERLEVEL\_SCIENTIST, [Engineer](#) = userLevelTypes::USERLEVEL\_ENGINEER }

## Signals

- void **setCurrentBoxIndex** (int index)

## 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.*
- **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)
- void **scaleBy** (const int m, const int d)
- [qcaobject::QCaObject](#) \* **createQcaltem** (unsigned int variableIndex)
- void **dragEnterEvent** (QDragEnterEvent \*event)
- void **dropEvent** (QDropEvent \*event)
- void **mousePressEvent** (QMouseEvent \*event)
- void **saveConfiguration** ([PersistianceManager](#) \*pm)
- void **restoreConfiguration** ([PersistianceManager](#) \*pm, [restorePhases](#) restorePhase)
- QString **copyVariable** ()
- QVariant **copyData** ()
- void **paste** (QVariant s)
- void **setDrop** (QVariant drop)
- QVariant **getDrop** ()



## 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](#)

### 9.98.1 Member Enumeration Documentation

#### 9.98.1.1 enum QEPvProperties::UserLevels

User friendly enumerations for [userLevelVisibility](#) and [userLevelEnabled](#) properties - refer to [userLevelVisibility](#) and [userLevelEnabled](#) properties and [userLevel](#) enumeration for details.

#### Enumerator:

- User** Refer to [USERLEVEL\\_USER](#) for details.
- Scientist** Refer to [USERLEVEL\\_SCIENTIST](#) for details.
- Engineer** Refer to [USERLEVEL\\_ENGINEER](#) for details.

### 9.98.2 Member Function Documentation

#### 9.98.2.1 void QEPvProperties::restoreConfiguration ( [PersistenceManager](#) \*, [restorePhases](#) ) [protected, virtual]

Service a request to restore the QE widget's configuration. A QE widget recover any configuration details from the [PersistenceManager](#). For example, a [QEStripChart](#) may restore the variables being plotted. Many QE widgets do not have any persistent 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](#).

#### 9.98.2.2 void QEPvProperties::saveConfiguration ( [PersistenceManager](#) \* ) [protected, virtual]

Service a request to save the QE widget's current configuration. A widget may save any configuration details through the [PersistenceManager](#). For example, a [QEStripChart](#)

may save the variables being plotted. Many QE widgets do not have any persistent data requirements and do not implement this method.

Reimplemented from [QEWidget](#).

**9.98.2.3** `void QEPvProperties::scaleBy ( const int , const int )` [protected, 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 from [QEWidget](#).

### 9.98.3 Property Documentation

**9.98.3.1** `bool QEPvProperties::allowDrop` [read, write]

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

Reimplemented from [QEDragDrop](#).

**9.98.3.2** `bool QEPvProperties::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](#).

**9.98.3.3** `unsigned QEPvProperties::int` [read, write]

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

**9.98.3.4** `UserLevels QEPvProperties::userLevelEnabled` [read, write]

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

mode is set application wide through the [QELogin](#) widget, or programatically through `setUserLevel()` Widgets that are always 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'.

#### 9.98.3.5 QString QEPvProperties::userLevelEngineerStyle [read, write]

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

#### 9.98.3.6 QString QEPvProperties::userLevelScientistStyle [read, write]

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

#### 9.98.3.7 QString QEPvProperties::userLevelUserStyle [read, write]

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

#### 9.98.3.8 UserLevels QEPvProperties::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through `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'.

#### 9.98.3.9 QString QEPvProperties::variable [read, write]

EPICS variable name (CA PV)

### 9.98.3.10 `bool QEPvProperties::variableAsToolTip` [read, write]

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

Reimplemented from [QEToolTip](#).

### 9.98.3.11 `QString QEPvProperties::variableSubstitutions` [read, write]

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

### 9.98.3.12 `bool QEPvProperties::visible` [read, write]

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

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

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

## 9.99 QEPvPropertiesManager Class Reference

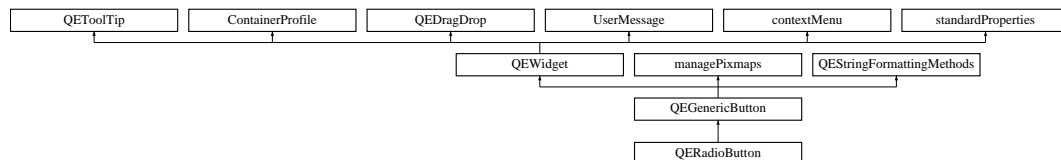
### Public Member Functions

- **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

Inheritance diagram for QERadioButton:



- enum `UserLevelTypes` { `USER` = `userLevelTypes::USERLEVEL_USER`, `Scientist` = `userLevelTypes::USERLEVEL_SCIENTIST`, `Engineer` = `userLevelTypes::USERLEVEL_ENGINEER` }
- enum `Formats` {  
`Default` = `QStringFormatting::FORMAT_DEFAULT`, `Floating` = `QStringFormatting::FORMAT_FLOATING`, `Integer` = `QStringFormatting::FORMAT_INTEGER`, `UnsignedInteger` = `QStringFormatting::FORMAT_UNSIGNEDINTEGER`,  
`Time` = `QStringFormatting::FORMAT_TIME`, `LocalEnumeration` = `QStringFormatting::FORMAT_LOCAL_ENUMERATE` }
- enum `Notations` { `Fixed` = `QStringFormatting::NOTATION_FIXED`, `Scientific` = `QStringFormatting::NOTATION_SCIENTIFIC`, `Automatic` = `QStringFormatting::NOTATION_AUTOMATIC` }
- enum `ArrayActions` { `Append` = `QStringFormatting::APPEND`, `Ascii` = `QStringFormatting::ASCII`, `Index` = `QStringFormatting::INDEX` }
- enum `UpdateOptions` { `Text` = `QGenericButton::UPDATE_TEXT`, `Icon` = `QGenericButton::UPDATE_ICON`, `TextAndIcon` = `QGenericButton::UPDATE_TEXT_AND_ICON`, `State` = `QGenericButton::UPDATE_STATE` }  
*User friendly enumerations for updateOption property - refer to QGenericButton::updateOptions for details.*
- enum `CreationOptionNames` { `Open` = `QForm::CREATION_OPTION_OPEN`, `NewTab` = `QForm::CREATION_OPTION_NEW_TAB`, `NewWindow` = `QForm::CREATION_OPTION_NEW_WINDOW` }

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

- void **launchGui** (QString guiName, QForm::creationOptions creationOption)

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

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

- void [newGui](#) (QString guiName, QForm::creationOptions creationOption)  
*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)

## 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](#)
- bool [writeOnPress](#)
- bool [writeOnRelease](#)
- bool [writeOnClick](#)
- QString [pressText](#)
- QString [releaseText](#)
- QString [clickText](#)
- QString [clickCheckedText](#)
- QString [labelText](#)
- QString [program](#)
- QStringList [arguments](#)
- QString [guiFile](#)
- [CreationOptionNames](#) [creationOption](#)
- QString [prioritySubstitutions](#)

### 9.100.1 Member Enumeration Documentation

#### 9.100.1.1 enum `QERadioButton::ArrayActions`

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

**Enumerator:**

**Append** Refer to [QCStringFormatting::APPEND](#) for details.

**Ascii** Refer to [QCStringFormatting::ASCII](#) for details.

**Index** Refer to [QCStringFormatting::INDEX](#) for details.

#### 9.100.1.2 enum `QERadioButton::CreationOptionNames`

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

**Enumerator:**

**Open** Replace the current GUI with the new GUI.

**NewTab** Open new GUI in a new tab.

**NewWindow** Open new GUI in a new window.

#### 9.100.1.3 enum `QERadioButton::Formats`

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

#### 9.100.1.4 enum `QERadioButton::Notations`

User friendly enumerations for notation property - refer to [QCStringFormatting::notations](#) for details.

**Enumerator:**

**Fixed** Refer to [QCStringFormatting::NOTATION\\_FIXED](#) for details.

**Scientific** Refer to [QCStringFormatting::NOTATION\\_SCIENTIFIC](#) for details.

**Automatic** Refer to [QCStringFormatting::NOTATION\\_AUTOMATIC](#) for details.



## 9.100.1.5 enum QERadioButton::UpdateOptions

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

**Enumerator:**

**Text** Data updates will update the button text.

**Icon** Data updates will update the button icon.

**TextAndIcon** Data updates will update the button text and icon.

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

## 9.100.1.6 enum QERadioButton::UserLevels

User friendly enumerations for [userLevelVisibility](#) and [userLevelEnabled](#) properties - refer to [userLevelVisibility](#) and [userLevelEnabled](#) properties and userLevel enumeration for details.

**Enumerator:**

**User** Refer to USERLEVEL\_USER for details.

**Scientist** Refer to USERLEVEL\_SCIENTIST for details.

**Engineer** Refer to USERLEVEL\_ENGINEER for details.

## 9.100.2 Constructor &amp; Destructor Documentation

9.100.2.1 QERadioButton::QERadioButton ( QWidget \* *parent* = 0 )

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

9.100.2.2 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.

## 9.100.3 Member Function Documentation

9.100.3.1 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)

### 9.100.3.2 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.

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

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

Reimplemented from [QEGenericButton](#).

### 9.100.3.4 void QERadioButton::pressed ( int value ) [signal]

Button has been Pressed. The value emitted is the integer interpretation of the press-Text property

### 9.100.3.5 void QERadioButton::released ( int value ) [signal]

Button has been Released The value emitted is the integer interpretation of the release-Text property

## 9.100.4 Property Documentation

### 9.100.4.1 bool QERadioButton::addUnits [read, write]

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

### 9.100.4.2 Qt::Alignment QERadioButton::alignment [read, write]

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

#### 9.100.4.3 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](#).

#### 9.100.4.4 QStringList QERadioButton::arguments [read, write]

Arguments for program specified in the 'program' property.

Reimplemented from [QEGenericButton](#).

#### 9.100.4.5 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'.

#### 9.100.4.6 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](#).

#### 9.100.4.7 QString QERadioButton::clickText [read, write]

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

Reimplemented from [QEGenericButton](#).

#### 9.100.4.8 `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

#### 9.100.4.9 `CreationOptionNames QERadioButton::creationOption` [read, write]

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

Reimplemented from [QEGenericButton](#).

#### 9.100.4.10 `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](#).

#### 9.100.4.11 `Formats QERadioButton::format` [read, write]

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

#### 9.100.4.12 `QString QERadioButton::guiFile` [read, write]

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

#### 9.100.4.13 `unsigned QERadioButton::int` [read, write]

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

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

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

#### 9.100.4.14 QString QERadioButton::labelText [read, write]

Button label text (prior to substitution). Macro substitutions will be applied to this text and the result will be set as the button text. Used when data updates are not being represented in the button text. 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](#).

#### 9.100.4.15 bool QERadioButton::leadingZero [read, write]

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

#### 9.100.4.16 QString QERadioButton::localEnumeration [read, write]

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

Format is:

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

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

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

may be included in a string to indicate a line break

Examples are:

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

The data value is converted to a string if no enumeration for that value is available. For example, if the local enumeration is '0:off,1:on', and a value of 10 is processed, the

text generated is '10'. If a blank string is required, this should be explicit. for example, '0:off,1:on,10:'"

A range of numbers can be covered by a pair of values as in the following example:  $\geq 4$ :"Between 4 and 8",  $\leq 8$ :"Between 4 and 8"

#### 9.100.4.17 Notations `QERadioButton::notation` [read, write]

Notation used for numerical formatting. Default is fixed.

#### 9.100.4.18 `QString QERadioButton::password` [read, write]

Password user will need to enter before any action is taken

Reimplemented from [QEGenericButton](#).

#### 9.100.4.19 `QPixmap QERadioButton::pixmap0` [read, write]

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

#### 9.100.4.20 `QPixmap QERadioButton::pixmap1` [read, write]

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

#### 9.100.4.21 `QPixmap QERadioButton::pixmap2` [read, write]

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

#### 9.100.4.22 `QPixmap QERadioButton::pixmap3` [read, write]

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

#### 9.100.4.23 `QPixmap QERadioButton::pixmap4` [read, write]

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

#### 9.100.4.24 `QPixmap QERadioButton::pixmap5` [read, write]

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

**9.100.4.25 QPixmap QERadioButton::pixmap6** [read, write]

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

**9.100.4.26 QPixmap QERadioButton::pixmap7** [read, write]

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

**9.100.4.27 int QERadioButton::precision** [read, write]

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

**9.100.4.28 QString QERadioButton::pressText** [read, write]

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

Reimplemented from [QEGenericButton](#).

**9.100.4.29 QString QERadioButton::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 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 substittions applied to the new form, they are appended to the list and the existing macro substitutions take precedence.

Reimplemented from [QEGenericButton](#).

**9.100.4.30 QString QERadioButton::program** [read, write]

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

Reimplemented from [QEGenericButton](#).

**9.100.4.31 QString QERadioButton::releaseText** [read, write]

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

Reimplemented from [QEGenericButton](#).

**9.100.4.32 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](#).

**9.100.4.33 bool QERadioButton::trailingZeros** [read, write]

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

**9.100.4.34 UpdateOptions QERadioButton::updateOption** [read, write]

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

Reimplemented from [QEGenericButton](#).

**9.100.4.35 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.

**9.100.4.36 UserLevels QERadioButton::userLevelEnabled** [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through `setUserLevel()` Widgets that are always 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'.

**9.100.4.37 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.

**9.100.4.38 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.

#### 9.100.4.39 `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.

#### 9.100.4.40 `UserLevels QERadioButton::userLevelVisibility` [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through `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'.

#### 9.100.4.41 `QString QERadioButton::variable` [read, write]

EPICS variable name (CA PV)

#### 9.100.4.42 `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](#).

#### 9.100.4.43 `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.

#### 9.100.4.44 `bool QERadioButton::visible` [read, write]

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

#### 9.100.4.45 `bool QERadioButton::writeOnClick` [read, write]

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

Reimplemented from [QEGenericButton](#).

#### 9.100.4.46 `bool QERadioButton::writeOnPress` [read, write]

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

Reimplemented from [QEGenericButton](#).

#### 9.100.4.47 `bool QERadioButton::writeOnRelease` [read, write]

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

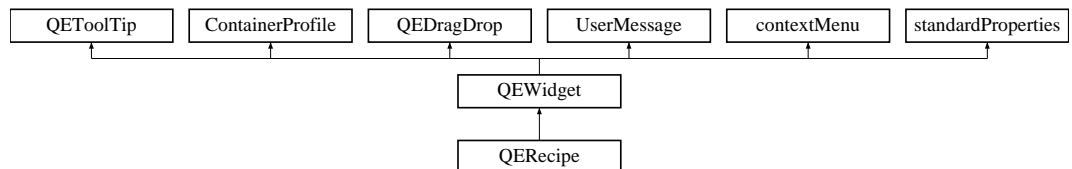
Reimplemented from [QEGenericButton](#).

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

## 9.101 QERecipe Class Reference

Inheritance diagram for QERecipe:



### Public Types

- enum **configurationTypesProperty** { **File** = FROM\_FILE, **Text** = FROM\_TEXT }
- enum **detailsLayoutProperty** { **Top** = TOP, **Bottom** = BOTTOM, **Left** = LEFT, **Right** = RIGHT }
- enum **userTypesProperty** { **User** = 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 **setDetailsLayout** (int pValue)
- int **getDetailsLayout** ()
- void **setCurrentUserType** (int pValue)
- int **getCurrentUserType** ()
- bool **saveRecipeList** ()
- void **refreshRecipeList** ()
- void **refreshButton** ()
- void **userLevelChanged** ([userLevelTypes::userLevels](#) pValue)
- void **setConfigurationTypeProperty** (configurationTypesProperty pConfigurationType)
- configurationTypesProperty **getConfigurationTypeProperty** ()
- void **setDetailsLayoutProperty** (detailsLayoutProperty pDetailsLayout)
- detailsLayoutProperty **getDetailsLayoutProperty** ()
- void **setCurrentUserTypeProperty** (userTypesProperty pUserType)
- userTypesProperty **getCurrentUserTypeProperty** ()

### Protected Attributes

- QLabel \* **qLabelRecipeDescription**
- QComboBox \* **qComboBoxRecipeList**
- QPushButton \* **qPushButtonNew**
- QPushButton \* **qPushButtonSave**
- QPushButton \* **qPushButtonDelete**
- QPushButton \* **qPushButtonApply**
- QPushButton \* **qPushButtonRead**
- [QEConfiguredLayout](#) \* **qEConfiguredLayoutRecipeFields**
- QDomDocument **document**
- QString **recipeFile**
- QString **filename**
- int **detailsLayout**
- int **currentUserType**

### Properties

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

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

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

## 9.102 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

## 9.103 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

## 9.104 QERecordSpecList Class Reference

### Public Member Functions

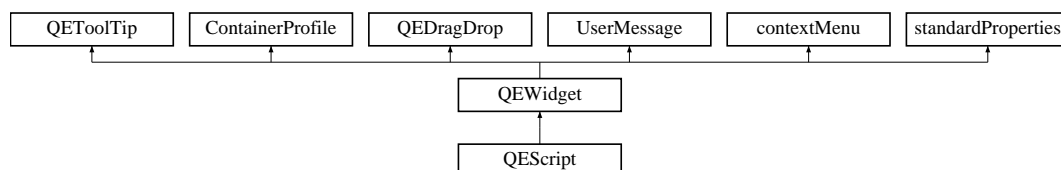
- [QERecordSpec](#) \* **find** (const QString recordType)
- void **appendOrReplace** ([QERecordSpec](#) \*recordSpec)
- bool **processRecordSpecFile** (const QString &filename)

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

## 9.105 QEScript Class Reference

Inheritance diagram for QEScript:



## Public Types

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

## Signals

- void **selected** (QString pFilename)

## Public Member Functions

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

## Protected Attributes

- QLineEdit \* **qlineEditDirectoryPath**
- QPushButton \* **qPushButtonDirectoryBrowser**
- QPushButton \* **qPushButtonRefresh**
- [\\_QTableWidgetScript](#) \* **qTableWidgetScript**
- QString **fileFilter**
- bool **showFileExtension**
- int **detailsLayout**

## Properties

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

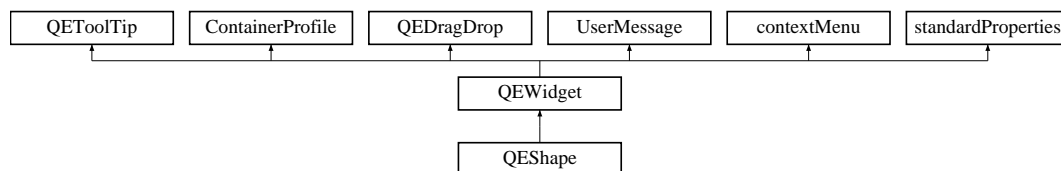
The documentation for this class was generated from the following files:

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

## 9.106 QEShape Class Reference

```
#include <QEShape.h>
```

Inheritance diagram for QEShape:



## Public Types

- enum [shapeOptions](#) {  
**Line, Points, Polyline, Polygon,**  
**Rect, RoundedRect, Ellipse, Arc,**  
**Chord, Pie, Path** }
- 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 qlonglong &out)
- void [dbValueChanged2](#) (const qlonglong &out)
- void [dbValueChanged3](#) (const qlonglong &out)
- void [dbValueChanged4](#) (const qlonglong &out)
- void [dbValueChanged5](#) (const qlonglong &out)
- void [dbValueChanged6](#) (const qlonglong &out)

## Public Member Functions

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

### 9.106.1 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 vertcal and horizontal size and position. It is tightly integrated with the base class [QEWidget](#) which provides generic support such as macro substitutions, drag/drop, and standard properties.

### 9.106.2 Member Enumeration Documentation

#### 9.106.2.1 enum QEShape::animationOptions

Options for how a variable will animate the shape.

#### 9.106.2.2 enum QEShape::shapeOptions

Options for the type of shape.

#### 9.106.2.3 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.

**9.106.3 Constructor & Destructor Documentation****9.106.3.1 QEShape::QEShape ( QWidget \* *parent* = 0 )**

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

**9.106.3.2 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.

**9.106.4 Member Function Documentation****9.106.4.1 void QEShape::dbValueChanged1 ( const qlonglong & *out* ) [signal]**

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

**9.106.4.2 void QEShape::dbValueChanged2 ( const qlonglong & *out* ) [signal]**

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

**9.106.4.3 void QEShape::dbValueChanged3 ( const qlonglong & *out* ) [signal]**

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

**9.106.4.4 void QEShape::dbValueChanged4 ( const qlonglong & *out* ) [signal]**

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

**9.106.4.5** void QEShape::dbValueChanged5 ( const qlonglong & *out* ) [signal]

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

**9.106.4.6** void QEShape::dbValueChanged6 ( const qlonglong & *out* ) [signal]

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

**9.106.5** Property Documentation**9.106.5.1** 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](#).

**9.106.5.2** 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.

**9.106.5.3** 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.

**9.106.5.4** 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.

**9.106.5.5** 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.

**9.106.5.6** 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.

**9.106.5.7 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.

**9.106.5.8 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'.

**9.106.5.9 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'.

**9.106.5.10 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'.

**9.106.5.11 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'.

**9.106.5.12 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'.

**9.106.5.13 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'.

**9.106.5.14 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'.

**9.106.5.15 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'.

**9.106.5.16 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'.

**9.106.5.17 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'.

**9.106.5.18 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](#).

**9.106.5.19 unsigned QEShape::int** [read, write]

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

The number of points to use when drawing shapes that are defined by a variable number of points, such as polyline, polygon, path, and series of points.

Sets the width of the pen. Used for the following shapes: Line, Points, Polyline, Polygon, Rect, RoundedRect, Ellipse, Arc, Chord, Pie, Path

**9.106.5.20 double QEShape::offset1** [read, write]

Offset applied to data from the 1st variable before it is used to animate the shape

**9.106.5.21 double QEShape::offset2** [read, write]

Offset applied to data from the 2nd variable before it is used to animate the shape

**9.106.5.22 double QEShape::offset3** [read, write]

Offset applied to data from the 3rd variable before it is used to animate the shape

**9.106.5.23 double QEShape::offset4** [read, write]

Offset applied to data from the 4th variable before it is used to animate the shape

**9.106.5.24 double QEShape::offset5** [read, write]

Offset applied to data from the 5th variable before it is used to animate the shape

**9.106.5.25 double QEShape::offset6** [read, write]

Offset applied to data from the 6th variable before it is used to animate the shape

**9.106.5.26 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

**9.106.5.27 QPoint QEShape::point10** [read, write]

10th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

**9.106.5.28 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

**9.106.5.29 QPoint QEShape::point3** [read, write]

3rd coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

**9.106.5.30 QPoint QEShape::point4** [read, write]

4th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

**9.106.5.31 QPoint QEShape::point5** [read, write]

5th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path



**9.106.5.32** `QPoint QEShape::point6` [read, write]

6th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

**9.106.5.33** `QPoint QEShape::point7` [read, write]

7th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

**9.106.5.34** `QPoint QEShape::point8` [read, write]

8th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

**9.106.5.35** `QPoint QEShape::point9` [read, write]

9th coordinate used when drawing the shape. Used for the following shapes: Points, Polyline, Polygon, Path

**9.106.5.36** `double QEShape::scale2` [read, write]

Scale factor applied to data from the 2nd variable before it is used to animate the shape

**9.106.5.37** `double QEShape::scale3` [read, write]

Scale factor applied to data from the 3rd variable before it is used to animate the shape

**9.106.5.38** `double QEShape::scale4` [read, write]

Scale factor applied to data from the 4th variable before it is used to animate the shape

**9.106.5.39** `double QEShape::scale5` [read, write]

Scale factor applied to data from the 5th variable before it is used to animate the shape

**9.106.5.40** `double QEShape::scale6` [read, write]

Scale factor applied to data from the 6th variable before it is used to animate the shape

#### 9.106.5.41 UserLevels QEShape::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through `setUserLevel()`. Widgets that are always 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'.

#### 9.106.5.42 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.

#### 9.106.5.43 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.

#### 9.106.5.44 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.

#### 9.106.5.45 UserLevels QEShape::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through `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'.

**9.106.5.46** `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`.

**9.106.5.47** `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`.

**9.106.5.48** `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`.

**9.106.5.49** `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`.

**9.106.5.50** `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`.

**9.106.5.51** `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`.

**9.106.5.52** `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](#).



- 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** ()

### 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](#)

### 9.107.1 Member Enumeration Documentation

#### 9.107.1.1 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.

### 9.107.2 Member Function Documentation

#### 9.107.2.1 void QESlider::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.

### 9.107.3 Member Data Documentation

#### 9.107.3.1 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).

### 9.107.4 Property Documentation

#### 9.107.4.1 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](#).

#### 9.107.4.2 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](#).

#### 9.107.4.3 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.

#### 9.107.4.4 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](#).

#### 9.107.4.5 UserLevels QESlider::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through `setUserLevel()` Widgets that are always 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'.

#### 9.107.4.6 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.

#### 9.107.4.7 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.

#### 9.107.4.8 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.

#### 9.107.4.9 UserLevels QESlider::userLevelVisibility [read, write]

Lowest user level at which the widget is visible. Default is 'User'. Used when designing GUIs that display more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through 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'.

#### 9.107.4.10 QString QESlider::variable [read, write]

EPICS variable name (CA PV)

#### 9.107.4.11 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](#).

#### 9.107.4.12 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.

#### 9.107.4.13 bool QESlider::visible [read, write]

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

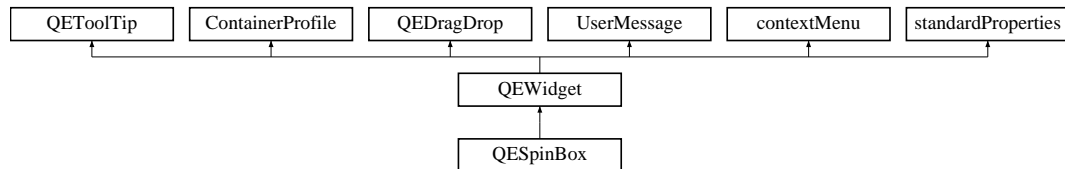
The documentation for this class was generated from the following files:

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



## 9.108 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** ()

### 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**

## 9.108.1 Member Enumeration Documentation

### 9.108.1.1 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.

**9.108.2 Member Function Documentation****9.108.2.1 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.

**9.108.3 Property Documentation****9.108.3.1 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](#).

**9.108.3.2 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](#).

**9.108.3.3 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.

**9.108.3.4 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](#).

#### 9.108.3.5 UserLevels QESpinBox::userLevelEnabled [read, write]

Lowest user level at which the widget is enabled. Default is 'User'. Used when designing GUIs that allow access to more and more detail according to the user mode. The user mode is set application wide through the [QELogin](#) widget, or programatically through `setUserLevel()`. Widgets that are always 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'.

#### 9.108.3.6 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.

#### 9.108.3.7 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.

#### 9.108.3.8 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.

#### 9.108.3.9 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'.

### 9.108.3.10 QString QESpinBox::variable [read, write]

EPICS variable name (CA PV)

### 9.108.3.11 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](#).

### 9.108.3.12 QString QESpinBox::variableSubstitutions [read, write]

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

### 9.108.3.13 bool QESpinBox::visible [read, write]

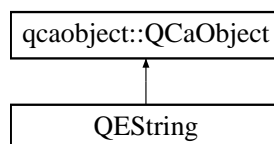
Display the widget. Default is true. Setting this property false is usefull if widget is only used to provide a signal - for example, when supplying data to a [QELink](#) widget. Note, when false the widget will still be visible in Qt Designer.

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

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

## 9.109 QEStrng Class Reference

Inheritance diagram for QEStrng:



### 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

- **QString** (QString recordName, QObject \*eventObject, [QStringFormatting](#) \*stringFormattingIn, unsigned int variableIndexIn)
- **QString** (QString recordName, QObject \*eventObject, [QStringFormatting](#) \*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/QCString.h
- /tmp/epicsqt/trunk/framework/data/src/QCString.cpp

## 9.110 QStringFormatting 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** ()

### 9.110.1 Member Enumeration Documentation

#### 9.110.1.1 enum QStringFormatting::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.

#### 9.110.1.2 enum QStringFormatting::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.

### 9.110.1.3 enum `QStringFormatting::notations`

Notations when formatting a floating point number

#### Enumerator:

***NOTATION\_FIXED*** Standard floating point 123456.789.

***NOTATION\_SCIENTIFIC*** Scientific representation 1.23456789e6.

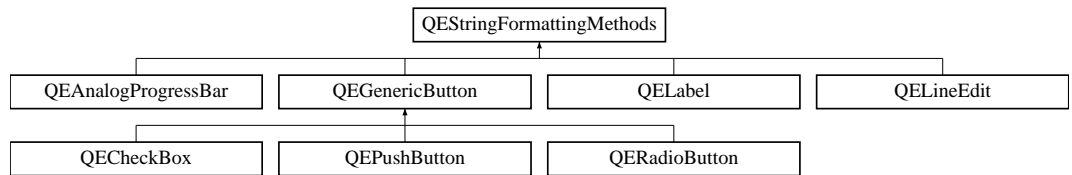
***NOTATION\_AUTOMATIC*** Automatic choice of standard or scientific notation.

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

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

## 9.111 `QStringFormattingMethods` Class Reference

Inheritance diagram for `QStringFormattingMethods`:



### 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** ([QStringFormatting::formats](#) format)
- [QStringFormatting::formats](#) **getFormat** ()
- void **setRadix** (unsigned int radix)
- unsigned int **getRadix** ()



- void **setNotation** ([QEStrFormatting::notations](#) notation)
- [QEStrFormatting::notations](#) **getNotation** ()
- void **setArrayAction** ([QEStrFormatting::arrayActions](#) arrayAction)
- [QEStrFormatting::arrayActions](#) **getArrayAction** ()
- void **setArrayIndex** (unsigned int arrayIndex)
- unsigned int **getArrayIndex** ()

### Protected Attributes

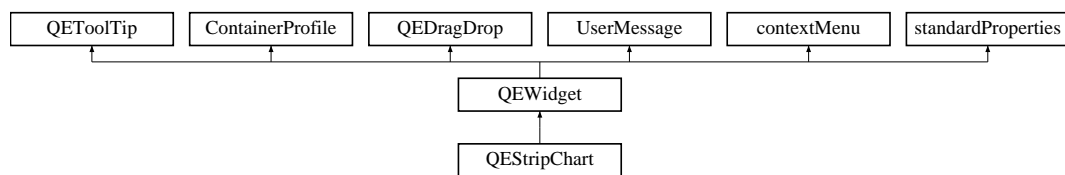
- [QEStrFormatting](#) **stringFormatting**

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

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

## 9.112 QEStripChart Class Reference

Inheritance diagram for QEStripChart:



### Classes

- class [PrivateData](#)

### Public Types

- enum **Constants** { **NUMBER\_OF\_PVS** = 12 }

### Public Member Functions

- **QEStripChart** (QWidget \*parent=0)
- QSize **sizeHint** () const
- QDateTime **getStartDateTime** ()
- QDateTime **getEndDateTime** ()
- void **setEndDateTime** (QDateTime endDateTimeIn)
- int **getDuration** ()

- void **setDuration** (int durationIn)
- double **getYMinimum** ()
- void **setYMinimum** (const double yMinimumIn)
- double **getYMaximum** ()
- void **setYMaximum** (const double yMaximumIn)
- void **setYRange** (const double yMinimumIn, const double yMaximumIn)
- void **plotData** ()
- void **addToPredefinedList** (const QString &pvName)
- QString **getPredefinedItem** (int i)

### Protected Member Functions

- void **dragEnterEvent** (QDragEnterEvent \*event)
- void **dropEvent** (QDropEvent \*event)
- void **setDrop** (QVariant drop)
- void **setup** ()
- [qcaobject::QCaObject](#) \* **createQcaltem** (unsigned int variableIndex)
- void **establishConnection** (unsigned int variableIndex)
- void [saveConfiguration](#) ([PersistenceManager](#) \*pm)
- void [restoreConfiguration](#) ([PersistenceManager](#) \*pm, [restorePhases](#) restorePhase)

### 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 **PrivateData**
- class [QEStripChartItem](#)

### 9.112.1 Member Function Documentation

9.112.1.1 void [QEStripChart::restoreConfiguration](#) ( [PersistenceManager](#) \*,  
restorePhases ) [protected, virtual]

Service a request to restore the QE widget's configuration. A QE widget recover any configuration details from the [PersistenceManager](#). For example, a [QEStripChart](#) may restore the variables being plotted. Many QE widgets do not have any persistent 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](#).

9.112.1.2 void [QEStripChart::saveConfiguration](#) ( [PersistenceManager](#) \* )  
[protected, virtual]

Service a request to save the QE widget's current configuration. A widget may save any configuration details through the [PersistenceManager](#). For example, a [QEStripChart](#) may save the variables being plotted. Many QE widgets do not have any persistent data requirements and do not implement this method.

Reimplemented from [QEWidget](#).

### 9.112.2 Property Documentation

9.112.2.1 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

## 9.113 QEStripChartAdjustPVDialo Class Reference

### Public Member Functions

- **QEStripChartAdjustPVDialo** (QWidget \*parent=0)
- void **setValueScaling** (const [ValueScaling](#) &valueScale)
- [ValueScaling](#) **getValueScaling** ()
- void **setSupport** (const double min, const double max, const [TrackRange](#) &lo-prHopr, const [TrackRange](#) &plotted, const [TrackRange](#) &buffered)

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartAdjustPVDialo.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartAdjustPVDialo.cpp

## 9.114 QEStripChartContextMenu Class Reference

### Signals

- void **contextMenuSelected** (const unsigned int, 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 QEStripChartNames::How how)
- void **setLineDrawMode** (const QEStripChartNames::LineDrawModes mode)
- QAction \* **exec** (const unsigned int slot, const QPoint &pos, QAction \*at=0)

### Static Public Attributes

- static const int **numberPrefefinedItems** = 10

### 9.114.1 Constructor & Destructor Documentation

9.114.1.1 **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](#)

## 9.115 QEStripChartItem Class Reference

### Classes

- class [PrivateData](#)

### Public Slots

- void **setColour** (const QColor &colour)

### Signals

- void **itemContextMenuRequested** (const unsigned int, const QPoint &)

### Public Member Functions

- **QEStripChartItem** ([QEStripChart](#) \*chart, QLabel \*pvName, [QELabel](#) \*caLabel, unsigned int slot)
- bool **isInUse** ()
- void **setPvName** (QString pvName, QString substitutions)
- QString **getPvName** ()
- void **setScaling** (const double d, const double m, const double c)
- void **getScaling** (double &d, double &m, double &c)
- bool **isScaled** ()
- bool **getUseReceiveTime** ()
- QArchiveInterface::How **getArchiveReadHow** ()
- QEStripChartNames::LineDrawModes **getLineDrawMode** ()
- QColor **getColour** ()
- [TrackRange](#) **getLoprHopr** (bool doScale)
- [TrackRange](#) **getDisplayedMinMax** (bool doScale)
- [TrackRange](#) **getBufferedMinMax** (bool doScale)
- void **readArchive** ()
- void **normalise** ()
- void **plotData** (const double timeScale, const QEStripChartNames::YScaleModes yScaleMode)
- void **contextMenuSelected** (const QEStripChartNames::ContextMenuOptions option)
- 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

## 9.116 QESTripChartItemDialog Class Reference

### Public Member Functions

- **QESTripChartItemDialog** (QWidget \*parent=0)
- void **setPvName** (QString pvNameIn)
- QString **getPvName** ()
- bool **isClear** ()

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

- /tmp/epicsqt/trunk/framework/widgets/QESTripChart/QESTripChartItemDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QESTripChart/QESTripChartItemDialog.cpp

## 9.117 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 **ContextMenuItemFirst** = **SCCM\_READ\_ARCHIVE**
- static const ContextMenuOptions **ContextMenuItemLast** = **SCCM\_PREDEFINED\_10**

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartNames.h

## 9.118 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

## 9.119 QEStripChartTimeDialog Class Reference

### Public Member Functions

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

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

- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartTimeDialog.h
- /tmp/epicsqt/trunk/framework/widgets/QEStripChart/QEStripChartTimeDialog.cpp

## 9.120 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 **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)

### 9.120.1 Detailed Description

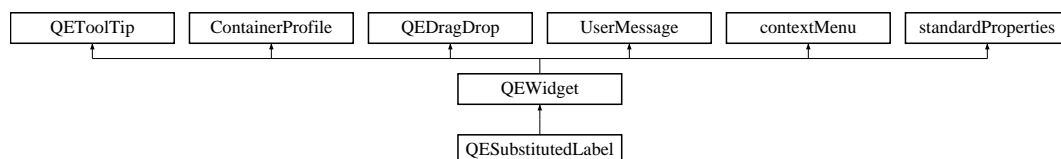
This class holds all the StripChart tool bar widgets.

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

## 9.121 QESubstitutedLabel Class Reference

Inheritance diagram for QESubstitutedLabel:



## Public Member Functions

- **QESubstitutedLabel** (QWidget \*parent=0)
- void **establishConnection** (unsigned int variableIndex)
- void **setLabelTextProperty** (QString labelTextIn)
- QString **getLabelTextProperty** ()
- QString **getLabelTextPropertyFormat** ()
- void **setLabelTextPropertyFormat** (QString labelTextIn)

### Protected Attributes

- QString [labelText](#)

### Properties

- QString [textSubstitutions](#)

#### 9.121.1 Member Data Documentation

9.121.1.1 **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

#### 9.121.2 Property Documentation

9.121.2.1 **QString QESubstitutedLabel::textSubstitutions** [read, write]

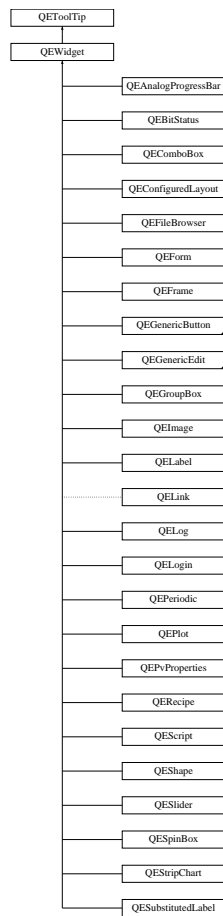
Text substitutions. These substitutions are applied to the 'labelText' property prior to copying it to the label text.

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

### 9.122 QEToolTip Class Reference

Inheritance diagram for QEToolTip:



## Public Member Functions

- **QEToolTip** (QWidget \*ownerIn)
- void **updateToolTipVariable** (const QString &variable)
- void **updateToolTipAlarm** (const QString &alarm)
- void **updateToolTipCustom** (const QString &custom)
- void **updateToolTipConnection** (bool connection)
- void **setVariableAsToolTip** (bool variableAsToolTip)
- bool **getVariableAsToolTip** ()

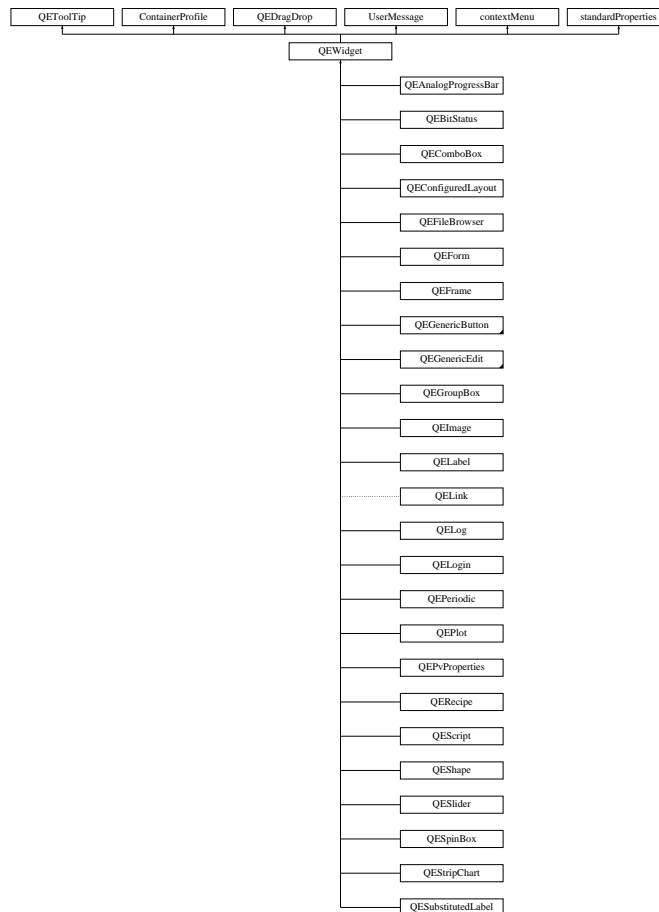
The documentation for this class was generated from the following files:

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

## 9.123 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 persistent 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 [getMessageSourceId](#) ()
- void [setMessageSourceId](#) (unsigned int messageId)
- [qcaobject::QCaObject](#) \* [getQcaltem](#) (unsigned int variableIndex)
- void [setUpContextMenu](#) (QWidget \*w)
- QColor [getColor](#) (QCaAlarmInfo &alarmInfo, const int saturation)
- void [processAlarmInfo](#) (QCaAlarmInfo &alarmInfo)
- 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](#) (PersistenceManager \*)
- virtual void [restoreConfiguration](#) (PersistenceManager \*, restorePhases)
- virtual void [scaleBy](#) (const int, const int)

### Static Public Member Functions

- static QFile \* [findQEFile](#) (QString name, ContainerProfile \*profile)
- static bool [inDesigner](#) ()

### Protected Member Functions

- void [setNumVariables](#) (unsigned int numVariablesIn)
- [qcaobject::QCaObject](#) \* [createConnection](#) (unsigned int variableIndex)
- virtual [qcaobject::QCaObject](#) \* [createQcaltem](#) (unsigned int variableIndex)
- virtual void [establishConnection](#) (unsigned int variableIndex)
- QString [persistantName](#) (QString prefix)

### Protected Attributes

- bool [subscribe](#)

#### 9.123.1 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 `createQcaltem`. A [QELabel](#), for example, wants string updates so it creates a [QEStrng](#) 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 it's `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 [QEStrng](#) object. The [QEStrng](#) 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 [QEStrng](#) object to its `setLabelText` slot.

## 9.123.2 Member Function Documentation

### 9.123.2.1 void QEWidget::activate ( )

Initiate updates. Called after all configuration is complete.

### 9.123.2.2 void QEWidget::deactivate ( )

Terminates updates. This has been provided for third party (non QEGui) applications using the framework.

### 9.123.2.3 QString QEWidget::defaultFileLocation ( )

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

### 9.123.2.4 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.

### 9.123.2.5 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

### 9.123.2.6 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.

### 9.123.2.7 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.

### 9.123.2.8 qcaobject::QCaObject \* QEWidget::getQcaltem ( unsigned int *variableIndex* )

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

### 9.123.2.9 `QFile * QEWidget::openQEFile ( QString name, QFile::OpenModeFlag mode )`

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

### 9.123.2.10 `void QEWidget::processAlarmInfo ( QCaAlarmInfo & alarmInfo )`

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 useful for single-variable widgets.

### 9.123.2.11 `void QEWidget::readNow ( )`

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

### 9.123.2.12 `virtual void QEWidget::restoreConfiguration ( PersistenceManager *, restorePhases ) [inline, virtual]`

Service a request to restore the QE widget's configuration. A QE widget recover any configuration details from the [PersistenceManager](#). For example, a [QEStripChart](#) may restore the variables being plotted. Many QE widgets do not have any persistent 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](#).

### 9.123.2.13 `virtual void QEWidget::saveConfiguration ( PersistenceManager * ) [inline, virtual]`

Service a request to save the QE widget's current configuration. A widget may save any configuration details through the [PersistenceManager](#). For example, a [QEStripChart](#) may save the variables being plotted. Many QE widgets do not have any persistent data requirements and do not implement this method.

Reimplemented in [QEPvProperties](#), and [QEStripChart](#).

### 9.123.2.14 `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 [QEPvProperties](#), and [QEShape](#).

9.123.2.15 void [QWidget::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.

9.123.2.16 void [QWidget::setupContextMenu](#) ( QWidget \* w )

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

9.123.2.17 void [QWidget::setVariableNameAndSubstitutions](#) ( QString *variableNameIn*,  
QString *variableNameSubstitutionsIn*, unsigned int *variableIndex* ) [virtual]

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

Reimplemented in [QEBitStatus](#), and [QEForm](#).

9.123.2.18 virtual void [QWidget::writeNow](#) ( ) [inline, virtual]

(Control widgets only - such as [QELineEdit](#)) Write the value now. Used when writeOn-Change, 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/QWidget.h
- /tmp/epicsqt/trunk/framework/widgets/src/QWidget.cpp

## 9.124 QWidgets Class Reference

### Public Member Functions

- **QWidgets** (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

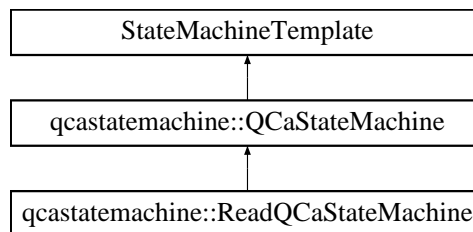
## 9.125 QLabelList Class Reference

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

- /tmp/epicsqt/trunk/framework/widgets/QEPvProperties/QEPvProperties.cpp

## 9.126 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

## 9.127 ROlinfo Class Reference

### Public Member Functions

- void **setX** (long x)
- void **setY** (long y)
- void **setW** (long w)
- void **setH** (long h)
- void **clearX** ()
- void **clearY** ()
- void **clearW** ()
- void **clearH** ()
- bool **getStatus** ()
- QRect **getArea** ()

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

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

## 9.128 SaveRestoreSignal Class Reference

### Public Types

- enum **saveRestoreOptions** { **SAVE**, **RESTORE\_APPLICATION**, **RESTORE\_QEFRAMEWORK** }

### Signals

- void **saveRestore** (SaveRestoreSignal::saveRestoreOptions option)

### Public Member Functions

- void **setOwner** ([PersistenceManager](#) \*ownerIn)
- void **save** ()
- void **restore** ()

### 9.128.1 Member Function Documentation

#### 9.128.1.1 void SaveRestoreSignal::restore ( )

!! signal must be blocking

#### 9.128.1.2 void SaveRestoreSignal::save ( )

!! signal must be blocking

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

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

## 9.129 saveRestoreSlot 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

## 9.130 selectMenu Class Reference

### Public Member Functions

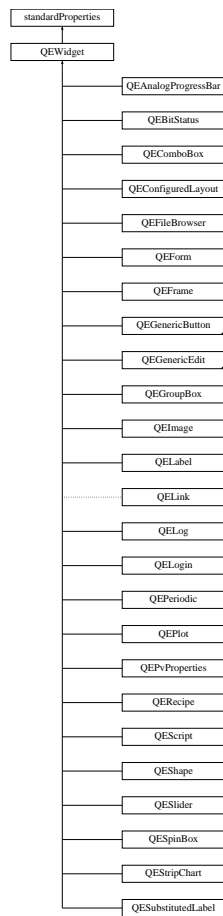
- **selectMenu** (QWidget \*parent=0)
- [imageContextMenu::imageContextMenuOptions](#) **getSelectOption** (const QPoint &pos)
- void **setChecked** (const int mode)
- void **setPanEnabled** (bool enablePan)
- void **setVSliceEnabled** (bool enableVSliceSelection)
- void **setHSliceEnabled** (bool enableHSliceSelection)
- void **setAreaEnabled** (bool enableAreaSelection)
- void **setProfileEnabled** (bool enableProfileSelection)
- void **setTargetEnabled** (bool enableTargetSelection)

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

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

## 9.131 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

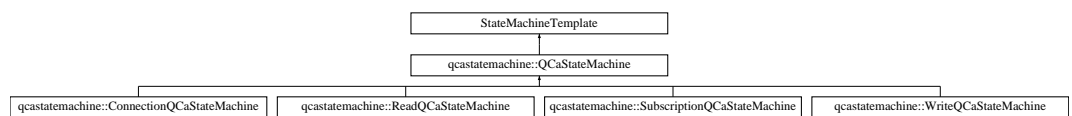
- void **checkVisibilityEnabledLevel** ([userLevelTypes::userLevels](#) level)

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

### 9.132 StateMachineTemplate Class Reference

Inheritance diagram for StateMachineTemplate:



#### Public Member Functions

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

#### Public Attributes

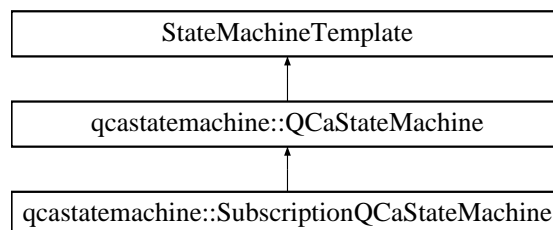
- int **currentState**
- int **requestState**

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

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

### 9.133 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

## 9.134 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

## 9.135 TrackRange Class Reference

### Public Member Functions

- void **clear** ()
- void **merge** (const double d)
- void **merge** (const [TrackRange](#) &that)
- bool **getMinMax** (double &min, double &max) const

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

## 9.136 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

## 9.137 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

## 9.138 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



## 9.139 `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

## 9.140 `userLevelTypes` Class Reference

### Public Types

- enum `userLevels` { `USERLEVEL_USER`, `USERLEVEL_SCIENTIST`, `USERLEVEL_ENGINEER` }

### 9.140.1 Member Enumeration Documentation

#### 9.140.1.1 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 privileged.

**`USERLEVEL_SCIENTIST`** User level - more privileged than user, less than engineer.

**`USERLEVEL_ENGINEER`** User level - most privileged.

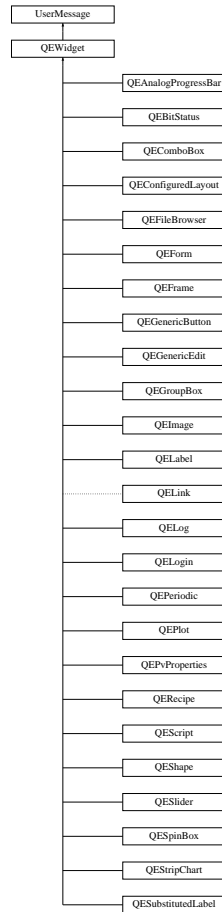
The documentation for this class was generated from the following file:

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

## 9.141 `UserMessage` Class Reference

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

Inheritance diagram for UserMessage:



## Public Types

- enum **message\_filter\_options** { MESSAGE\_FILTER\_ANY, MESSAGE\_FILTER\_MATCH, MESSAGE\_FILTER\_NONE }

## Public Member Functions

- void **setSourceId** (unsigned int sourceId)  
*Set the source ID (the ID set up by the GUI designer, usually matched to the source ID of logging widgets)*
- void **setFormId** (unsigned int formId)  
*Set the form ID (the the same ID for all sibling widgets within an [QEForm](#) widget)*
- void **setFormFilter** (message\_filter\_options formFilterIn)  
*Set the message filtering applied to the form ID.*
- void **setSourceFilter** (message\_filter\_options sourceFilterIn)

- Set the message filtering applied to the source ID.*

  - unsigned int [getSourceId](#) ()
  - Get the source ID (the ID set up by the GUI designer, usually matched to the source ID of logging widgets).*
  - unsigned int [getFormId](#) ()
  - Get the form ID (the the same ID for all sibling widgets within an [QEForm](#) widget)*
  - message\_filter\_options [getFormFilter](#) ()
  - Get the message filtering applied to the form ID.*
  - message\_filter\_options [getSourceFilter](#) ()
  - Get the message filtering applied to the source ID.*
  - void [setChildFormId](#) (unsigned int)
  - Set the for ID of all widgets that are children of this widget.*
  - unsigned int [getChildFormId](#) ()
  - Get the for ID of all widgets that are children of this widget.*
  - unsigned int [getNextMessageFormId](#) ()
  - Generate a new form ID for all widgets in a new form.*
  - void [sendMessage](#) (QString message, [message\\_types](#) type=[message\\_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](#)

### 9.141.1 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

## 9.142 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.*

### 9.142.1 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

## 9.143 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.*

### 9.143.1 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

## 9.144 ValueScaling Class Reference

### Public Member Functions

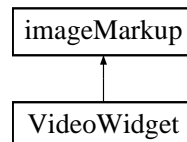
- void **reset** ()
- void **assign** (const [ValueScaling](#) &s)
- void **set** (const double dIn, const double mIn, const double cIn)
- void **get** (double &dOut, double &mOut, double &cOut)
- void **map** (const double fromLower, const double fromUpper, const double toLower, const double toUpper)
- bool **isScaled** ()
- double **value** (const double x)
- [TrackRange](#) **value** (const [TrackRange](#) &x)
- void **saveConfiguration** ([PMElement](#) &parentElement)
- 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

## 9.145 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)

### 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)
- int **scaleImageOrdinate** (int ord)
- QImage **getImage** ()

### Protected Member Functions

- void **paintEvent** (QPaintEvent \*)
- void **mousePressEvent** (QMouseEvent \*event)
- void **mouseReleaseEvent** (QMouseEvent \*event)
- void **mouseMoveEvent** (QMouseEvent \*event)
- void **wheelEvent** (QWheelEvent \*event)
- void **markupChange** (QImage &markups, QVector< QRect > &changedAreas)
- void **resizeEvent** (QResizeEvent \*event)
- void **markupSetCursor** (QCursor cursor)
- void **markupAction** (markupIds mode, bool 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/QEImage/videowidget.h
- /tmp/epicsqt/trunk/framework/widgets/QEImage/videowidget.cpp

## 9.146 WidgetRef Class Reference

### Public Member Functions

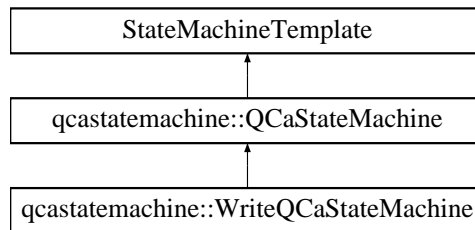
- **WidgetRef** (QEWidget \*refIn)
- QEWidget \* **getRef** ()

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

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

## 9.147 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

## 9.148 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

- [\\_Field, 27](#)
- [\\_Item, 28](#)
- [\\_QDialogItem, 28](#)
- [\\_QPushButtonGroup, 28](#)
- [\\_QTableWidgetFileBrowser, 29](#)
- [\\_QTableWidgetLog, 29](#)
- [\\_QTableWidgetScript, 29](#)
- activate
  - [QEObject, 265](#)
- addUnits
  - [QEObjectProgressbar, 72](#)
  - [QEObject, 88](#)
  - [QEObject, 150](#)
  - [QEObjectEdit, 158](#)
  - [QEObjectNumericEdit, 171](#)
  - [QEObjectPushButton, 193](#)
  - [QEObjectRadioButton, 212](#)
- alarmSeverityDisplayMode
  - [QEObjectProgressbar, 72](#)
- alignment
  - [QEObject, 88](#)
  - [QEObjectPushButton, 193](#)
  - [QEObjectRadioButton, 212](#)
- allowDrop
  - [QEObjectProgressbar, 72](#)
  - [QEObjectStatus, 79](#)
  - [QEObject, 88](#)
  - [QEObjectComboBox, 99](#)
  - [QEObjectFrame, 111](#)
  - [QEObjectGenericEdit, 118](#)
  - [QEObjectGroupBox, 123](#)
  - [QEObjectImage, 135](#)
  - [QEObjectLabel, 150](#)
  - [QEObjectPeriodic, 176](#)
  - [QEObjectPlot, 185](#)
  - [QEObjectPushButton, 193](#)
  - [QEObjectPvProperties, 204](#)
  - [QEObjectRadioButton, 212](#)
  - [QEObjectShape, 231](#)
  - [QEObjectSlider, 240](#)
  - [QEObjectSpinBox, 245](#)
- altReadbackVariable
  - [QEObjectPushButton, 193](#)
- animation1
  - [QEObjectShape, 231](#)
- animation2
  - [QEObjectShape, 231](#)
- animation3
  - [QEObjectShape, 231](#)
- animation4
  - [QEObjectShape, 231](#)
- animation5
  - [QEObjectShape, 231](#)
- animation6
  - [QEObjectShape, 231](#)
- animationOptions
  - [QEObjectShape, 229](#)
- APPEND
  - [QEObjectStringFormatting, 249](#)
- Append
  - [QEObjectAnalogProgressbar, 71](#)
  - [QEObject, 85](#)
  - [QEObjectLabel, 148](#)
  - [QEObjectLineEdit, 156](#)
  - [QEObjectPushButton, 190](#)
  - [QEObjectRadioButton, 210](#)
- areaColor
  - [QEObjectImage, 136](#)
- arguments
  - [QEObject, 88](#)
  - [QEObjectPushButton, 193](#)
  - [QEObjectRadioButton, 213](#)
- arrayAction
  - [QEObjectAnalogProgressbar, 73](#)
  - [QEObject, 88](#)
  - [QEObjectLabel, 150](#)
  - [QEObjectLineEdit, 158](#)
  - [QEObjectPushButton, 194](#)
  - [QEObjectRadioButton, 213](#)
- ArrayActions
  - [QEObjectAnalogProgressbar, 71](#)

- QCheckBox, [85](#)
  - QLabel, [148](#)
  - QLineEdit, [156](#)
  - QEPushButton, [190](#)
  - QERadioButton, [210](#)
- arrayActions
  - QStringFormatting, [249](#)
- ASCII
  - QStringFormatting, [249](#)
- Ascii
  - QAnalogProgressBar, [71](#)
  - QCheckBox, [85](#)
  - QLabel, [148](#)
  - QLineEdit, [156](#)
  - QEPushButton, [190](#)
  - QERadioButton, [210](#)
- autoBrightnessContrast
  - QImage, [135](#)
- Automatic
  - QAnalogProgressBar, [71](#)
  - QCheckBox, [86](#)
  - QLabel, [148](#)
  - QLineEdit, [157](#)
  - QEPushButton, [191](#)
  - QERadioButton, [210](#)
- autoScale
  - QNumericEdit, [171](#)
- backgroundColour
  - QAnalogIndicator, [66](#)
- Bar
  - QAnalogIndicator, [66](#)
- beamColor
  - QImage, [136](#)
- beamXVariable
  - QImage, [136](#)
- beamYVariable
  - QImage, [136](#)
- borderColour
  - QAnalogIndicator, [66](#)
- Bottom\_To\_Top
  - QAnalogIndicator, [66](#)
- centreAngle
  - QAnalogIndicator, [66](#)
- ChartState, [30](#)
- clickCheckedText
  - QCheckBox, [88](#)
  - QEPushButton, [194](#)
  - QERadioButton, [213](#)
- clicked
  - QCheckBox, [87](#)
  - QEPushButton, [192](#)
  - QERadioButton, [211](#)
- clickText
  - QCheckBox, [89](#)
  - QEPushButton, [194](#)
  - QERadioButton, [213](#)
- clippingHighVariable
  - QImage, [136](#)
- clippingLowVariable
  - QImage, [136](#)
- clippingOnOffVariable
  - QImage, [136](#)
- color1
  - QShape, [232](#)
- color10
  - QShape, [232](#)
- color2
  - QShape, [232](#)
- color3
  - QShape, [232](#)
- color4
  - QShape, [232](#)
- color5
  - QShape, [232](#)
- color6
  - QShape, [232](#)
- color7
  - QShape, [232](#)
- color8
  - QShape, [232](#)
- color9
  - QShape, [233](#)
- confirmAction
  - QCheckBox, [89](#)
  - QEPushButton, [194](#)
  - QERadioButton, [214](#)
- confirmWrite
  - QGenericEdit, [119](#)
- ContainerProfile, [31](#)
- contextMenu, [33](#)
- contextMenuObject, [35](#)
- creationOption
  - QCheckBox, [89](#)
  - QEPushButton, [194](#)
  - QERadioButton, [214](#)
- CreationOptionNames
  - QCheckBox, [85](#)
  - QEPushButton, [190](#)

- QERadioButton, 210
- dbElementChanged
  - QEPeriodic, 175
- dbValueChanged
  - QEAnalogProgressBar, 72
  - QEBitStatus, 78
  - QECheckBox, 87
  - QECombobox, 98
  - QEImage, 135
  - QELabel, 150
  - QELineEdit, 157
  - QENumericEdit, 170
  - QEPeriodic, 175
  - QEPlot, 184
  - QEPushButton, 192
  - QERadioButton, 211
  - QESlider, 240
  - QESpinBox, 245
- dbValueChanged1
  - QEShape, 230
- dbValueChanged2
  - QEShape, 230
- dbValueChanged3
  - QEShape, 230
- dbValueChanged4
  - QEShape, 230
- dbValueChanged5
  - QEShape, 230
- dbValueChanged6
  - QEShape, 231
- deactivate
  - QEWWidget, 265
- Default
  - QEAnalogProgressBar, 71
  - QECheckBox, 85
  - QELabel, 148
  - QELineEdit, 157
  - QEPushButton, 191
  - QERadioButton, 210
- defaultFileLocation
  - QEWWidget, 265
- displayAlarmState
  - QEAnalogProgressBar, 73
  - QEBitStatus, 79
  - QECheckBox, 89
  - QECombobox, 99
  - QEFrame, 111
  - QEGenericEdit, 119
  - QEGroupBox, 123
- QEImage, 136
- QELabel, 150
- QEPeriodic, 176
- QEPlot, 185
- QEPushButton, 195
- QEPvProperties, 204
- QERadioButton, 214
- QEShape, 233
- QESlider, 240
- QESpinBox, 245
- displayButtonBar
  - QEImage, 135
- drawMarkup
  - markupHLine, 42
  - markupVLine, 48
- enableBrightnessContrast
  - QEImage, 135
- enableHozSliceSelection
  - QEImage, 136
- enableVertSliceSelection
  - QEImage, 137
- Engineer
  - QEAnalogProgressBar, 72
  - QEBitStatus, 78
  - QECheckBox, 86
  - QECombobox, 98
  - QEFrame, 111
  - QEGenericEdit, 117
  - QEGroupBox, 122
  - QEImage, 134
  - QELabel, 149
  - QEPeriodic, 175
  - QEPlot, 184
  - QEPushButton, 192
  - QEPvProperties, 203
  - QERadioButton, 211
  - QEShape, 230
  - QESlider, 240
  - QESpinBox, 245
- findQEFile
  - QEWWidget, 265
- Fit
  - QEImage, 133
- Fixed
  - QEAnalogProgressBar, 71
  - QECheckBox, 86
  - QELabel, 148
  - QELineEdit, 157

- QEPushButton, 191
- QERadioButton, 210
- flipRotateMenu, 36
- Floating
  - QEAnalogProgressBar, 71
  - QECheckBox, 85
  - QELabel, 148
  - QELineEdit, 157
  - QEPushButton, 191
  - QERadioButton, 210
- floating
  - QCaDateTime, 58
- fontColour
  - QEAnalogIndicator, 66
- foregroundColour
  - QEAnalogIndicator, 67
- format
  - QEAnalogProgressBar, 73
  - QECheckBox, 89
  - QELabel, 150
  - QELineEdit, 158
  - QEPushButton, 195
  - QERadioButton, 214
- FORMAT\_DEFAULT
  - QStringFormatting, 249
- FORMAT\_FLOATING
  - QStringFormatting, 249
- FORMAT\_INTEGER
  - QStringFormatting, 249
- FORMAT\_LOCAL\_ENUMERATE
  - QStringFormatting, 249
- FORMAT\_STRING
  - QStringFormatting, 249
- FORMAT\_TIME
  - QStringFormatting, 249
- FORMAT\_UNSIGNEDINTEGER
  - QStringFormatting, 249
- formatInteger
  - QEIntegerFormatting, 144
- formatIntegerArray
  - QEIntegerFormatting, 144
- formatOption
  - QEImage, 137
- FormatOptions
  - QEImage, 132
- formatOptions
  - QEImage, 132
- Formats
  - QEAnalogProgressBar, 71
  - QECheckBox, 85
  - QELabel, 148
  - QELineEdit, 156
  - QEPushButton, 191
  - QERadioButton, 210
- formats
  - QStringFormatting, 249
- formatValue
  - QEIntegerFormatting, 144
- getColor
  - QEWidget, 265
- getConfirmWrite
  - QEGenericEdit, 117
- getElement
  - PMElementList, 52
- getFrameworkVersion
  - QEWidget, 265
- getLocalEnumeration
  - QELocalEnumeration, 163
- getMessageSourceId
  - QEWidget, 265
- getQcaltm
  - QEWidget, 265
- getSubscribe
  - QEGenericEdit, 117
- getWriteOnEnter
  - QEGenericEdit, 117
- getWriteOnFinish
  - QEGenericEdit, 118
- getWriteOnLoseFocus
  - QEGenericEdit, 118
- GREY12
  - QEImage, 132
- GREY16
  - QEImage, 132
- GREY8
  - QEImage, 132
- Grey\_12
  - QEImage, 133
- Grey\_16
  - QEImage, 133
- Grey\_8
  - QEImage, 133
- guiFile
  - QECheckBox, 90
  - QEPushButton, 195
  - QERadioButton, 214
- heightVariable
  - QEImage, 137

- horizontalFlip
  - QImage, [137](#)
- hozSliceColor
  - QImage, [137](#)
- Icon
  - QCheckBox, [86](#)
  - QEPushButton, [191](#)
  - QERadioButton, [211](#)
- imageContextMenu, [36](#)
- imageInfo, [37](#)
- imageMarkup, [38](#)
- imageVariable
  - QImage, [137](#)
- INDEX
  - QStringFormatting, [249](#)
- Index
  - QAnalogProgressBar, [71](#)
  - QCheckBox, [85](#)
  - QLabel, [148](#)
  - QLineEdit, [156](#)
  - QEPushButton, [190](#)
  - QERadioButton, [210](#)
- initialHosScrollPos
  - QImage, [137](#)
- initialVertScrollPos
  - QImage, [135](#)
- int
  - QAnalogProgressBar, [73](#)
  - QBitStatus, [79](#)
  - QCheckBox, [90](#)
  - QComboBox, [99](#)
  - QFrame, [111](#)
  - QGenericEdit, [119](#)
  - QGroupBox, [123](#)
  - QImage, [137](#)
  - QLabel, [151](#)
  - QLineEdit, [158](#)
  - QPeriodic, [176](#)
  - QEPPlot, [185](#)
  - QEPushButton, [195](#)
  - QEPvProperties, [204](#)
  - QERadioButton, [214](#)
  - QEShape, [233](#)
  - QESlider, [240](#)
  - QESpinBox, [245](#)
- Integer
  - QAnalogProgressBar, [71](#)
  - QCheckBox, [86](#)
  - QLabel, [148](#)
- QLineEdit, [157](#)
- QEPushButton, [191](#)
- QERadioButton, [210](#)
- isDefined
  - QELocalEnumeration, [163](#)
- labelText
  - QCheckBox, [90](#)
  - QEPushButton, [195](#)
  - QERadioButton, [215](#)
  - QESubstitutedLabel, [260](#)
- launchGui
  - QCheckBox, [87](#)
  - QEPushButton, [192](#)
  - QERadioButton, [212](#)
- leadingZero
  - QAnalogProgressBar, [74](#)
  - QCheckBox, [90](#)
  - QLabel, [151](#)
  - QLineEdit, [158](#)
  - QEPushButton, [196](#)
  - QERadioButton, [215](#)
- leadingZeros
  - QNumericEdit, [171](#)
- Left\_To\_Right
  - QAnalogIndicator, [66](#)
- LocalEnumeration
  - QAnalogProgressBar, [71](#)
  - QCheckBox, [86](#)
  - QLabel, [148](#)
  - QLineEdit, [157](#)
  - QEPushButton, [191](#)
  - QERadioButton, [210](#)
- localEnumeration
  - QAnalogProgressBar, [74](#)
  - QCheckBox, [90](#)
  - QComboBox, [99](#)
  - QLabel, [151](#)
  - QLineEdit, [158](#)
  - QEPushButton, [196](#)
  - QERadioButton, [215](#)
- loginWidget, [39](#)
- logScale
  - QAnalogIndicator, [67](#)
- logScaleInterval
  - QAnalogIndicator, [67](#)
- majorInterval
  - QAnalogIndicator, [67](#)
- managePixmap, [40](#)

- markupBeam, [40](#)
- markupHLine, [41](#)
  - drawMarkup, [42](#)
- markupItem, [42](#)
- markupLine, [44](#)
- markupRegion, [45](#)
- markupTarget, [45](#)
- markupText, [46](#)
- markupVLine, [47](#)
  - drawMarkup, [48](#)
- maximum
  - QEAnalogIndicator, [67](#)
  - QENumericEdit, [171](#)
- message\_types, [48](#)
- Meter
  - QEAnalogIndicator, [66](#)
- minimum
  - QEAnalogIndicator, [67](#)
  - QENumericEdit, [171](#)
- minorInterval
  - QEAnalogIndicator, [67](#)
- mode
  - QEAnalogIndicator, [67](#)
- Modes
  - QEAnalogIndicator, [66](#)
- NewTab
  - QECheckBox, [85](#)
  - QEPushButton, [191](#)
  - QERadioButton, [210](#)
- NewWindow
  - QECheckBox, [85](#)
  - QEPushButton, [191](#)
  - QERadioButton, [210](#)
- NoRotation
  - QEImage, [134](#)
- notation
  - QEAnalogProgressBar, [74](#)
  - QECheckBox, [91](#)
  - QELabel, [152](#)
  - QELineEdit, [159](#)
  - QEPushButton, [196](#)
  - QERadioButton, [216](#)
- NOTATION\_AUTOMATIC
  - QStringFormatting, [250](#)
- NOTATION\_FIXED
  - QStringFormatting, [250](#)
- NOTATION\_SCIENTIFIC
  - QStringFormatting, [250](#)
- Notations
  - QEAnalogProgressBar, [71](#)
  - QECheckBox, [86](#)
  - QELabel, [148](#)
  - QELineEdit, [157](#)
  - QEPushButton, [191](#)
  - QERadioButton, [210](#)
- notations
  - QStringFormatting, [249](#)
- offset1
  - QEShape, [233](#)
- offset2
  - QEShape, [233](#)
- offset3
  - QEShape, [233](#)
- offset4
  - QEShape, [233](#)
- offset5
  - QEShape, [234](#)
- offset6
  - QEShape, [234](#)
- Open
  - QECheckBox, [85](#)
  - QEPushButton, [191](#)
  - QERadioButton, [210](#)
- openQEFile
  - QEWidget, [265](#)
- orientation
  - QEAnalogIndicator, [67](#)
- Orientations
  - QEAnalogIndicator, [66](#)
- password
  - QECheckBox, [91](#)
  - QEPushButton, [197](#)
  - QERadioButton, [216](#)
- PeriodicDialog, [49](#)
- PeriodicElementSetupForm, [50](#)
- PeriodicSetupDialog, [50](#)
- PersistenceManager, [50](#)
- Picture
  - QELabel, [149](#)
- pixmap0
  - QECheckBox, [91](#)
  - QELabel, [152](#)
  - QEPushButton, [197](#)
  - QERadioButton, [216](#)
- pixmap1
  - QECheckBox, [91](#)
  - QELabel, [152](#)

- QEPushButton, [197](#)
- QERadioButton, [216](#)
- pixmap2
  - QCheckBox, [91](#)
  - QELabel, [152](#)
  - QEPushButton, [197](#)
  - QERadioButton, [216](#)
- pixmap3
  - QCheckBox, [92](#)
  - QELabel, [152](#)
  - QEPushButton, [197](#)
  - QERadioButton, [216](#)
- pixmap4
  - QCheckBox, [92](#)
  - QELabel, [152](#)
  - QEPushButton, [197](#)
  - QERadioButton, [216](#)
- pixmap5
  - QCheckBox, [92](#)
  - QELabel, [152](#)
  - QEPushButton, [197](#)
  - QERadioButton, [216](#)
- pixmap6
  - QCheckBox, [92](#)
  - QELabel, [152](#)
  - QEPushButton, [197](#)
  - QERadioButton, [216](#)
- pixmap7
  - QCheckBox, [92](#)
  - QELabel, [152](#)
  - QEPushButton, [197](#)
  - QERadioButton, [217](#)
- PMContext, [51](#)
- PMElement, [51](#)
- PMElementList, [52](#)
  - getElement, [52](#)
- point1
  - QEShape, [234](#)
- point10
  - QEShape, [234](#)
- point2
  - QEShape, [234](#)
- point3
  - QEShape, [234](#)
- point4
  - QEShape, [234](#)
- point5
  - QEShape, [234](#)
- point6
  - QEShape, [234](#)
- point7
  - QEShape, [235](#)
- point8
  - QEShape, [235](#)
- point9
  - QEShape, [235](#)
- precision
  - QEAAnalogProgressBar, [74](#)
  - QCheckBox, [92](#)
  - QELabel, [153](#)
  - QELineEdit, [159](#)
  - QENumericEdit, [171](#)
  - QEPushButton, [198](#)
  - QERadioButton, [217](#)
- pressed
  - QCheckBox, [87](#)
  - QEPushButton, [193](#)
  - QERadioButton, [212](#)
- pressText
  - QCheckBox, [92](#)
  - QEPushButton, [198](#)
  - QERadioButton, [217](#)
- prioritySubstitutions
  - QCheckBox, [92](#)
  - QEPushButton, [198](#)
  - QERadioButton, [217](#)
- processAlarmInfo
  - QEWidgit, [266](#)
- profileColor
  - QImage, [137](#)
- profilePlot, [53](#)
- program
  - QCheckBox, [93](#)
  - QEPushButton, [198](#)
  - QERadioButton, [217](#)
- PublishedProfile, [54](#)
- PushButtonSpecifications, [54](#)
- QBitStatus, [55](#)
- QCaAlarmInfo, [56](#)
- QCaConnectionInfo, [57](#)
- QCaDataPoint, [57](#)
- QCaDataPointList, [58](#)
- QCaDateTime, [58](#)
  - floating, [58](#)
- QCaEventFilter, [58](#)
- QCaEventItem, [59](#)
- QCaEventUpdate, [59](#)
- QCaInstalledFiltersListItem, [60](#)
- qcaobject::QCaObject, [60](#)

- qcastatemachine::ConnectionQCaStateMachine, displayAlarmState, 73
  - 31
- qcastatemachine::QCaStateMachine, 62
- qcastatemachine::ReadQCaStateMachine,
  - 268
- qcastatemachine::SubscriptionQCaStateMachine, Formats, 71
  - 272
- qcastatemachine::WriteQCaStateMachine,
  - 281
- QCaVariableNamePropertyManager, 62
- QEAnalogIndicator, 63
  - backgroundColour, 66
  - Bar, 66
  - borderColour, 66
  - Bottom\_To\_Top, 66
  - centreAngle, 66
  - fontColour, 66
  - foregroundColour, 67
  - Left\_To\_Right, 66
  - logScale, 67
  - logScaleInterval, 67
  - majorInterval, 67
  - maximum, 67
  - Meter, 66
  - minimum, 67
  - minorInterval, 67
  - mode, 67
  - Modes, 66
  - orientation, 67
  - Orientations, 66
  - Right\_To\_Left, 66
  - Scale, 66
  - showScale, 67
  - showText, 68
  - spanAngle, 68
  - Top\_To\_Bottom, 66
  - value, 68
- QEAnalogIndicator::Band, 30
- QEAnalogIndicator::BandList, 30
- QEAnalogProgressBar, 68
  - addUnits, 72
  - alarmSeverityDisplayMode, 72
  - allowDrop, 72
  - Append, 71
  - arrayAction, 73
  - ArrayActions, 71
  - Ascii, 71
  - Automatic, 71
  - dbValueChanged, 72
  - Default, 71
  - displayAlarmState, 73
    - Engineer, 72
    - Fixed, 71
    - Floating, 71
    - format, 73
  - Formats, 71
  - Index, 71
  - int, 73
  - Integer, 71
  - leadingZero, 74
  - LocalEnumeration, 71
  - localEnumeration, 74
  - notation, 74
  - Notations, 71
  - precision, 74
  - QEAnalogProgressBar, 72
  - Scientific, 71
  - Scientist, 72
  - Time, 71
  - trailingZeros, 74
  - UnsignedInteger, 71
  - useDbDisplayLimits, 75
  - useDbPrecision, 75
  - User, 72
  - userLevelEnabled, 75
  - userLevelEngineerStyle, 75
  - UserLevels, 71
  - userLevelScientistStyle, 75
  - userLevelUserStyle, 75
  - userLevelVisibility, 76
  - variable, 76
  - variableAsToolTip, 76
  - variableSubstitutions, 76
  - visible, 76
- QEBitStatus, 77
  - allowDrop, 79
  - dbValueChanged, 78
  - displayAlarmState, 79
  - Engineer, 78
  - int, 79
  - Scientist, 78
  - setVariableNameAndSubstitutions, 78
  - User, 78
  - userLevelEnabled, 79
  - userLevelEngineerStyle, 79
  - UserLevels, 78
  - userLevelScientistStyle, 80
  - userLevelUserStyle, 80
  - userLevelVisibility, 80
  - variable, 80



- variableAsToolTip, [80](#)
- variableSubstitutions, [80](#)
- visible, [81](#)
- QEByteArray, [81](#)
- QEChartStateLists, [82](#)
- QCheckBox, [82](#)
  - addUnits, [88](#)
  - alignment, [88](#)
  - allowDrop, [88](#)
  - Append, [85](#)
  - arguments, [88](#)
  - arrayAction, [88](#)
  - ArrayActions, [85](#)
  - Ascii, [85](#)
  - Automatic, [86](#)
  - clickCheckedText, [88](#)
  - clicked, [87](#)
  - clickText, [89](#)
  - confirmAction, [89](#)
  - creationOption, [89](#)
  - CreationOptionNames, [85](#)
  - dbValueChanged, [87](#)
  - Default, [85](#)
  - displayAlarmState, [89](#)
  - Engineer, [86](#)
  - Fixed, [86](#)
  - Floating, [85](#)
  - format, [89](#)
  - Formats, [85](#)
  - guiFile, [90](#)
  - Icon, [86](#)
  - Index, [85](#)
  - int, [90](#)
  - Integer, [86](#)
  - labelText, [90](#)
  - launchGui, [87](#)
  - leadingZero, [90](#)
  - LocalEnumeration, [86](#)
  - localEnumeration, [90](#)
  - NewTab, [85](#)
  - NewWindow, [85](#)
  - notation, [91](#)
  - Notations, [86](#)
  - Open, [85](#)
  - password, [91](#)
  - pixmap0, [91](#)
  - pixmap1, [91](#)
  - pixmap2, [91](#)
  - pixmap3, [92](#)
  - pixmap4, [92](#)
  - pixmap5, [92](#)
  - pixmap6, [92](#)
  - pixmap7, [92](#)
  - precision, [92](#)
  - pressed, [87](#)
  - pressText, [92](#)
  - prioritySubstitutions, [92](#)
  - program, [93](#)
  - QCheckBox, [87](#)
  - released, [87](#)
  - releaseText, [93](#)
  - Scientific, [86](#)
  - Scientist, [86](#)
  - State, [86](#)
  - subscribe, [93](#)
  - Text, [86](#)
  - TextAndIcon, [86](#)
  - Time, [86](#)
  - trailingZeros, [93](#)
  - UnsignedInteger, [86](#)
  - updateOption, [93](#)
  - UpdateOptions, [86](#)
  - useDbPrecision, [93](#)
  - User, [86](#)
  - userLevelEnabled, [93](#)
  - userLevelEngineerStyle, [94](#)
  - UserLevels, [86](#)
  - userLevelScientistStyle, [94](#)
  - userLevelUserStyle, [94](#)
  - userLevelVisibility, [94](#)
  - variable, [94](#)
  - variableAsToolTip, [95](#)
  - variableSubstitutions, [95](#)
  - visible, [95](#)
  - writeOnClick, [95](#)
  - writeOnPress, [95](#)
  - writeOnRelease, [95](#)
- QCheckBoxManager, [96](#)
- QComboBox, [96](#)
  - allowDrop, [99](#)
  - dbValueChanged, [98](#)
  - displayAlarmState, [99](#)
  - Engineer, [98](#)
  - int, [99](#)
  - localEnumeration, [99](#)
  - Scientist, [98](#)
  - subscribe, [99](#)
  - useDbEnumerations, [98](#)
  - User, [98](#)
  - userLevelEnabled, [99](#)

- userLevelEngineerStyle, 100
  - UserLevels, 98
  - userLevelScientistStyle, 100
  - userLevelUserStyle, 100
  - userLevelVisibility, 100
  - variable, 100
  - variableAsToolTip, 100
  - variableSubstitutions, 101
  - visible, 101
  - writeOnChange, 98
- QEConfiguredLayout, 101
- QEConfiguredLayoutManager, 103
- QEDragDrop, 103
- QEFileBrowser, 105
- QEFloating, 106
- QEFloatingFormatting, 107
- QEForm, 108
  - setVariableNameAndSubstitutions, 109
- QEFrame, 109
  - allowDrop, 111
  - displayAlarmState, 111
  - Engineer, 111
  - int, 111
  - Scientist, 111
  - User, 111
  - userLevelEnabled, 111
  - userLevelEngineerStyle, 111
  - UserLevels, 111
  - userLevelScientistStyle, 112
  - userLevelUserStyle, 112
  - userLevelVisibility, 112
  - variableAsToolTip, 112
  - visible, 112
- QEGenericButton, 113
- QEGenericEdit, 115
  - allowDrop, 118
  - confirmWrite, 119
  - displayAlarmState, 119
  - Engineer, 117
  - getConfirmWrite, 117
  - getSubscribe, 117
  - getWriteOnEnter, 117
  - getWriteOnFinish, 118
  - getWriteOnLoseFocus, 118
  - int, 119
  - QEGenericEdit, 117
  - Scientist, 117
  - setConfirmWrite, 118
  - setSubscribe, 118
  - setWriteOnEnter, 118
  - setWriteOnFinish, 118
  - setWriteOnLoseFocus, 118
  - subscribe, 119
  - User, 117
  - userLevelEnabled, 119
  - userLevelEngineerStyle, 119
  - UserLevels, 117
  - userLevelScientistStyle, 120
  - userLevelUserStyle, 120
  - userLevelVisibility, 120
  - variable, 120
  - variableAsToolTip, 120
  - variableSubstitutions, 120
  - visible, 121
  - writeOnEnter, 121
  - writeOnFinish, 121
  - writeOnLoseFocus, 121
- QEGroupBox, 121
  - allowDrop, 123
  - displayAlarmState, 123
  - Engineer, 122
  - int, 123
  - Scientist, 122
  - User, 122
  - userLevelEnabled, 123
  - userLevelEngineerStyle, 123
  - UserLevels, 122
  - userLevelScientistStyle, 123
  - userLevelUserStyle, 124
  - userLevelVisibility, 124
  - variableAsToolTip, 124
  - visible, 124
- QEImage, 125
  - allowDrop, 135
  - areaColor, 136
  - autoBrightnessContrast, 135
  - beamColor, 136
  - beamXVariable, 136
  - beamYVariable, 136
  - clippingHighVariable, 136
  - clippingLowVariable, 136
  - clippingOnOffVariable, 136
  - dbValueChanged, 135
  - displayAlarmState, 136
  - displayButtonBar, 135
  - enableBrightnessContrast, 135
  - enableHozSliceSelection, 136
  - enableVertSliceSelection, 137
  - Engineer, 134
  - Fit, 133

formatOption, [137](#)  
FormatOptions, [132](#)  
formatOptions, [132](#)  
GREY12, [132](#)  
GREY16, [132](#)  
GREY8, [132](#)  
Grey\_12, [133](#)  
Grey\_16, [133](#)  
Grey\_8, [133](#)  
heightVariable, [137](#)  
horizontalFlip, [137](#)  
hozSliceColor, [137](#)  
imageVariable, [137](#)  
initialHosScrollPos, [137](#)  
initialVertScrollPos, [135](#)  
int, [137](#)  
NoRotation, [134](#)  
profileColor, [137](#)  
QEImage, [134](#)  
regionOfInterest1HVariable, [138](#)  
regionOfInterest1WVariable, [138](#)  
regionOfInterest1XVariable, [138](#)  
regionOfInterest1YVariable, [138](#)  
regionOfInterest2HVariable, [138](#)  
regionOfInterest2WVariable, [138](#)  
regionOfInterest2XVariable, [138](#)  
regionOfInterest2YVariable, [138](#)  
regionOfInterest3HVariable, [138](#)  
regionOfInterest3WVariable, [139](#)  
regionOfInterest3XVariable, [139](#)  
regionOfInterest3YVariable, [139](#)  
regionOfInterest4HVariable, [139](#)  
regionOfInterest4WVariable, [139](#)  
regionOfInterest4XVariable, [139](#)  
regionOfInterest4YVariable, [139](#)  
RESIZE\_OPTION\_FIT, [133](#)  
RESIZE\_OPTION\_ZOOM, [133](#)  
resizeOption, [139](#)  
ResizeOptions, [133](#)  
resizeOptions, [133](#)  
RGB\_888, [132](#)  
Rotate180, [134](#)  
Rotate90Left, [134](#)  
Rotate90Right, [134](#)  
rotation, [139](#)  
ROTATION\_0, [133](#)  
ROTATION\_180, [133](#)  
ROTATION\_90\_LEFT, [133](#)  
ROTATION\_90\_RIGHT, [133](#)  
RotationOptions, [133](#)  
rotationOptions, [133](#)  
Scientist, [134](#)  
selectOptions, [134](#)  
showTime, [140](#)  
SO\_AREA4, [134](#)  
SO\_BEAM, [134](#)  
SO\_HSLICE, [134](#)  
SO\_NONE, [134](#)  
SO\_PANNING, [134](#)  
SO\_PROFILE, [134](#)  
SO\_TARGET, [134](#)  
SO\_VSLICE, [134](#)  
targetColor, [140](#)  
targetTriggerVariable, [140](#)  
targetXVariable, [140](#)  
targetYVariable, [140](#)  
timeColor, [140](#)  
User, [134](#)  
userLevelEnabled, [140](#)  
userLevelEngineerStyle, [140](#)  
UserLevels, [134](#)  
userLevelScientistStyle, [141](#)  
userLevelUserStyle, [141](#)  
userLevelVisibility, [141](#)  
variableAsToolTip, [141](#)  
variableSubstitutions, [141](#)  
verticalFlip, [142](#)  
vertSliceColor, [142](#)  
visible, [142](#)  
widthVariable, [142](#)  
Zoom, [133](#)  
QEInteger, [142](#)  
QEIntegerFormatting, [143](#)  
    formatInteger, [144](#)  
    formatIntegerArray, [144](#)  
    formatValue, [144](#)  
QELabel, [144](#)  
    addUnits, [150](#)  
    allowDrop, [150](#)  
    Append, [148](#)  
    arrayAction, [150](#)  
    ArrayActions, [148](#)  
    Ascii, [148](#)  
    Automatic, [148](#)  
    dbValueChanged, [150](#)  
    Default, [148](#)  
    displayAlarmState, [150](#)  
    Engineer, [149](#)  
    Fixed, [148](#)  
    Floating, [148](#)

- format, 150
- Formats, 148
- Index, 148
- int, 151
- Integer, 148
- leadingZero, 151
- LocalEnumeration, 148
- localEnumeration, 151
- notation, 152
- Notations, 148
- Picture, 149
- pixmap0, 152
- pixmap1, 152
- pixmap2, 152
- pixmap3, 152
- pixmap4, 152
- pixmap5, 152
- pixmap6, 152
- pixmap7, 152
- precision, 153
- QELabel, 149
- Scientific, 148
- Scientist, 149
- Text, 149
- Time, 148
- trailingZeros, 153
- UnsignedInteger, 148
- UPDATE\_PIXMAP, 149
- UPDATE\_TEXT, 149
- updateOption, 153
- UpdateOptions, 148
- updateOptions, 149
- useDbPrecision, 153
- User, 149
- userLevelEnabled, 153
- userLevelEngineerStyle, 153
- UserLevels, 149
- userLevelScientistStyle, 153
- userLevelUserStyle, 154
- userLevelVisibility, 154
- variable, 154
- variableAsToolTip, 154
- variableSubstitutions, 154
- visible, 154
- QELineEdit, 155
  - addUnits, 158
  - Append, 156
  - arrayAction, 158
  - ArrayActions, 156
  - Ascii, 156
  - Automatic, 157
  - dbValueChanged, 157
  - Default, 157
  - Fixed, 157
  - Floating, 157
  - format, 158
  - Formats, 156
  - Index, 156
  - int, 158
  - Integer, 157
  - leadingZero, 158
  - LocalEnumeration, 157
  - localEnumeration, 158
  - notation, 159
  - Notations, 157
  - precision, 159
  - QELineEdit, 157
  - Scientific, 157
  - Time, 157
  - trailingZeros, 159
  - UnsignedInteger, 157
  - useDbPrecision, 159
- QELineEditManager, 160
- QELink, 160
- QELocalEnumeration, 162
  - getLocalEnumeration, 163
  - isDefined, 163
  - QELocalEnumeration, 163
  - setLocalEnumeration, 163
  - textToDouble, 164
  - textToInt, 164
  - textToValue, 164
  - valueToText, 164
- QELog, 164
- QELogin, 167
- QELoginDialog, 167
- QENumericEdit, 168
  - addUnits, 171
  - autoScale, 171
  - dbValueChanged, 170
  - leadingZeros, 171
  - maximum, 171
  - minimum, 171
  - precision, 171
  - QENumericEdit, 170
- QENumericEditManager, 171
- QEPeriodic, 172
  - allowDrop, 176
  - dbElementChanged, 175
  - dbValueChanged, 175

- displayAlarmState, 176
- Engineer, 175
- int, 176
- readbackLabelVariable1, 176
- readbackLabelVariable2, 176
- Scientist, 175
- subscribe, 176
- User, 175
- userLevelEnabled, 177
- userLevelEngineerStyle, 177
- UserLevels, 175
- userLevelScientistStyle, 177
- userLevelUserStyle, 177
- userLevelVisibility, 177
- variableAsToolTip, 178
- variableSubstitutions, 178
- visible, 178
- writeButtonVariable1, 178
- writeButtonVariable2, 178
- QEPeiodic::elementInfoStruct, 35
- QEPeiodic::userInfoStructArray, 274
- QEPeiodicComponentData, 178
- QEPeiodicTaskMenu, 179
- QEPeiodicTaskMenuFactory, 179
- QEpicsPV, 180
- QEPlot, 181
  - allowDrop, 185
  - dbValueChanged, 184
  - displayAlarmState, 185
  - Engineer, 184
  - int, 185
  - Scientist, 184
  - User, 184
  - userLevelEnabled, 185
  - userLevelEngineerStyle, 185
  - UserLevels, 184
  - userLevelScientistStyle, 185
  - userLevelUserStyle, 186
  - userLevelVisibility, 186
  - variable1, 186
  - variable2, 186
  - variable3, 186
  - variable4, 186
  - variableAsToolTip, 187
  - variableSubstitutions, 187
  - visible, 187
- QEPushButton, 187
  - addUnits, 193
  - alignment, 193
  - allowDrop, 193
  - altReadbackVariable, 193
  - Append, 190
  - arguments, 193
  - arrayAction, 194
  - ArrayActions, 190
  - Ascii, 190
  - Automatic, 191
  - clickCheckedText, 194
  - clicked, 192
  - clickText, 194
  - confirmAction, 194
  - creationOption, 194
  - CreationOptionNames, 190
  - dbValueChanged, 192
  - Default, 191
  - displayAlarmState, 195
  - Engineer, 192
  - Fixed, 191
  - Floating, 191
  - format, 195
  - Formats, 191
  - guiFile, 195
  - Icon, 191
  - Index, 190
  - int, 195
  - Integer, 191
  - labelText, 195
  - launchGui, 192
  - leadingZero, 196
  - LocalEnumeration, 191
  - localEnumeration, 196
  - NewTab, 191
  - NewWindow, 191
  - notation, 196
  - Notations, 191
  - Open, 191
  - password, 197
  - pixmap0, 197
  - pixmap1, 197
  - pixmap2, 197
  - pixmap3, 197
  - pixmap4, 197
  - pixmap5, 197
  - pixmap6, 197
  - pixmap7, 197
  - precision, 198
  - pressed, 193
  - pressText, 198
  - prioritySubstitutions, 198
  - program, 198

- QEPushButton, 192
- released, 193
- releaseText, 198
- Scientific, 191
- Scientist, 192
- State, 191
- subscribe, 198
- Text, 191
- TextAndIcon, 191
- Time, 191
- trailingZeros, 198
- UnsignedInteger, 191
- updateOption, 199
- UpdateOptions, 191
- useDbPrecision, 199
- User, 192
- userLevelEnabled, 199
- userLevelEngineerStyle, 199
- UserLevels, 191
- userLevelScientistStyle, 199
- userLevelUserStyle, 199
- userLevelVisibility, 200
- variable, 200
- variableAsToolTip, 200
- variableSubstitutions, 200
- visible, 200
- writeOnClick, 200
- writeOnPress, 201
- writeOnRelease, 201
- QEPVNameLists, 201
- QEPvProperties, 201
  - allowDrop, 204
  - displayAlarmState, 204
  - Engineer, 203
  - int, 204
  - restoreConfiguration, 203
  - saveConfiguration, 203
  - scaleBy, 204
  - Scientist, 203
  - User, 203
  - userLevelEnabled, 204
  - userLevelEngineerStyle, 205
  - UserLevels, 203
  - userLevelScientistStyle, 205
  - userLevelUserStyle, 205
  - userLevelVisibility, 205
  - variable, 205
  - variableAsToolTip, 205
  - variableSubstitutions, 206
  - visible, 206
- QEPvProperties::OwnWidgets, 49
- QEPvPropertiesManager, 206
- QERadioButton, 207
  - addUnits, 212
  - alignment, 212
  - allowDrop, 212
  - Append, 210
  - arguments, 213
  - arrayAction, 213
  - ArrayActions, 210
  - Ascii, 210
  - Automatic, 210
  - clickCheckedText, 213
  - clicked, 211
  - clickText, 213
  - confirmAction, 214
  - creationOption, 214
  - CreationOptionNames, 210
  - dbValueChanged, 211
  - Default, 210
  - displayAlarmState, 214
  - Engineer, 211
  - Fixed, 210
  - Floating, 210
  - format, 214
  - Formats, 210
  - guiFile, 214
  - Icon, 211
  - Index, 210
  - int, 214
  - Integer, 210
  - labelText, 215
  - launchGui, 212
  - leadingZero, 215
  - LocalEnumeration, 210
  - localEnumeration, 215
  - NewTab, 210
  - NewWindow, 210
  - notation, 216
  - Notations, 210
  - Open, 210
  - password, 216
  - pixmap0, 216
  - pixmap1, 216
  - pixmap2, 216
  - pixmap3, 216
  - pixmap4, 216
  - pixmap5, 216
  - pixmap6, 216
  - pixmap7, 217

- precision, [217](#)
- pressed, [212](#)
- pressText, [217](#)
- prioritySubstitutions, [217](#)
- program, [217](#)
- QERadioButton, [211](#)
- released, [212](#)
- releaseText, [217](#)
- Scientific, [210](#)
- Scientist, [211](#)
- State, [211](#)
- subscribe, [217](#)
- Text, [211](#)
- TextAndIcon, [211](#)
- Time, [210](#)
- trailingZeros, [218](#)
- UnsignedInteger, [210](#)
- updateOption, [218](#)
- UpdateOptions, [210](#)
- useDbPrecision, [218](#)
- User, [211](#)
- userLevelEnabled, [218](#)
- userLevelEngineerStyle, [218](#)
- UserLevels, [211](#)
- userLevelScientistStyle, [218](#)
- userLevelUserStyle, [219](#)
- userLevelVisibility, [219](#)
- variable, [219](#)
- variableAsToolTip, [219](#)
- variableSubstitutions, [219](#)
- visible, [219](#)
- writeOnClick, [219](#)
- writeOnPress, [220](#)
- writeOnRelease, [220](#)
- QERecipe, [220](#)
- QERecordFieldName, [222](#)
- QERecordSpec, [223](#)
- QERecordSpecList, [223](#)
- QEScript, [223](#)
- QEShape, [225](#)
  - allowDrop, [231](#)
  - animation1, [231](#)
  - animation2, [231](#)
  - animation3, [231](#)
  - animation4, [231](#)
  - animation5, [231](#)
  - animation6, [231](#)
  - animationOptions, [229](#)
  - color1, [232](#)
  - color10, [232](#)
  - color2, [232](#)
  - color3, [232](#)
  - color4, [232](#)
  - color5, [232](#)
  - color6, [232](#)
  - color7, [232](#)
  - color8, [232](#)
  - color9, [233](#)
  - dbValueChanged1, [230](#)
  - dbValueChanged2, [230](#)
  - dbValueChanged3, [230](#)
  - dbValueChanged4, [230](#)
  - dbValueChanged5, [230](#)
  - dbValueChanged6, [231](#)
  - displayAlarmState, [233](#)
  - Engineer, [230](#)
  - int, [233](#)
  - offset1, [233](#)
  - offset2, [233](#)
  - offset3, [233](#)
  - offset4, [233](#)
  - offset5, [234](#)
  - offset6, [234](#)
  - point1, [234](#)
  - point10, [234](#)
  - point2, [234](#)
  - point3, [234](#)
  - point4, [234](#)
  - point5, [234](#)
  - point6, [234](#)
  - point7, [235](#)
  - point8, [235](#)
  - point9, [235](#)
  - QEShape, [230](#)
  - scale2, [235](#)
  - scale3, [235](#)
  - scale4, [235](#)
  - scale5, [235](#)
  - scale6, [235](#)
  - Scientist, [230](#)
  - shapeOptions, [229](#)
  - User, [230](#)
  - userLevelEnabled, [235](#)
  - userLevelEngineerStyle, [236](#)
  - UserLevels, [229](#)
  - userLevelScientistStyle, [236](#)
  - userLevelUserStyle, [236](#)
  - userLevelVisibility, [236](#)
  - variable1, [236](#)
  - variable2, [237](#)

- variable3, [237](#)
- variable4, [237](#)
- variable5, [237](#)
- variable6, [237](#)
- variableAsToolTip, [237](#)
- variableSubstitutions, [237](#)
- visible, [238](#)
- QESlider, [238](#)
  - allowDrop, [240](#)
  - dbValueChanged, [240](#)
  - displayAlarmState, [240](#)
  - Engineer, [240](#)
  - int, [240](#)
  - Scientist, [240](#)
  - subscribe, [241](#)
  - User, [240](#)
  - userLevelEnabled, [241](#)
  - userLevelEngineerStyle, [241](#)
  - UserLevels, [240](#)
  - userLevelScientistStyle, [241](#)
  - userLevelUserStyle, [241](#)
  - userLevelVisibility, [242](#)
  - variable, [242](#)
  - variableAsToolTip, [242](#)
  - variableSubstitutions, [242](#)
  - visible, [242](#)
  - writeOnChange, [240](#)
- QESpinBox, [243](#)
  - allowDrop, [245](#)
  - dbValueChanged, [245](#)
  - displayAlarmState, [245](#)
  - Engineer, [245](#)
  - int, [245](#)
  - Scientist, [245](#)
  - subscribe, [245](#)
  - User, [245](#)
  - userLevelEnabled, [245](#)
  - userLevelEngineerStyle, [246](#)
  - UserLevels, [244](#)
  - userLevelScientistStyle, [246](#)
  - userLevelUserStyle, [246](#)
  - userLevelVisibility, [246](#)
  - variable, [246](#)
  - variableAsToolTip, [247](#)
  - variableSubstitutions, [247](#)
  - visible, [247](#)
- QString, [247](#)
- QStringFormatting, [248](#)
  - APPEND, [249](#)
  - arrayActions, [249](#)
  - ASCII, [249](#)
  - FORMAT\_DEFAULT, [249](#)
  - FORMAT\_FLOATING, [249](#)
  - FORMAT\_INTEGER, [249](#)
  - FORMAT\_LOCAL\_ENUMERATE, [249](#)
  - FORMAT\_STRING, [249](#)
  - FORMAT\_TIME, [249](#)
  - FORMAT\_UNSIGNEDINTEGER, [249](#)
  - formats, [249](#)
  - INDEX, [249](#)
  - NOTATION\_AUTOMATIC, [250](#)
  - NOTATION\_FIXED, [250](#)
  - NOTATION\_SCIENTIFIC, [250](#)
  - notations, [249](#)
- QStringFormattingMethods, [250](#)
- QEStriptChart, [251](#)
  - restoreConfiguration, [253](#)
  - saveConfiguration, [253](#)
  - variableSubstitutions, [253](#)
- QEStriptChart::PrivateData, [52](#)
- QEStriptChartAdjustPVDialo, [254](#)
- QEStriptChartContextMenu, [254](#)
  - QEStriptChartContextMenu, [254](#)
- QEStriptChartItem, [255](#)
- QEStriptChartItem::PrivateData, [53](#)
- QEStriptChartItemDialog, [256](#)
- QEStriptChartNames, [256](#)
- QEStriptChartRangeDialog, [257](#)
- QEStriptChartTimeDialog, [258](#)
- QEStriptChartToolBar, [258](#)
- QEStriptChartToolBar::OwnWidgets, [48](#)
- QESubstitutedLabel, [259](#)
  - labelText, [260](#)
  - textSubstitutions, [260](#)
- QEToolTip, [260](#)
- QEWiget, [262](#)
  - activate, [265](#)
  - deactivate, [265](#)
  - defaultFileLocation, [265](#)
  - findQEFile, [265](#)
  - getColor, [265](#)
  - getFrameworkVersion, [265](#)
  - getMessageSourceId, [265](#)
  - getQcaltem, [265](#)
  - openQEFile, [265](#)
  - processAlarmInfo, [266](#)
  - readNow, [266](#)
  - restoreConfiguration, [266](#)
  - saveConfiguration, [266](#)
  - scaleBy, [266](#)



- setMessageSourceId, [267](#)
- setupContextMenu, [267](#)
- setVariableNameAndSubstitutions, [267](#)
- writeNow, [267](#)
- QEWidgets, [267](#)
- QLabelList, [268](#)
  
- readbackLabelVariable1
  - QEPeriodic, [176](#)
- readbackLabelVariable2
  - QEPeriodic, [176](#)
- readNow
  - QEWidget, [266](#)
- regionOfInterest1HVariable
  - QEImage, [138](#)
- regionOfInterest1WVariable
  - QEImage, [138](#)
- regionOfInterest1XVariable
  - QEImage, [138](#)
- regionOfInterest1YVariable
  - QEImage, [138](#)
- regionOfInterest2HVariable
  - QEImage, [138](#)
- regionOfInterest2WVariable
  - QEImage, [138](#)
- regionOfInterest2XVariable
  - QEImage, [138](#)
- regionOfInterest2YVariable
  - QEImage, [138](#)
- regionOfInterest3HVariable
  - QEImage, [138](#)
- regionOfInterest3WVariable
  - QEImage, [139](#)
- regionOfInterest3XVariable
  - QEImage, [139](#)
- regionOfInterest3YVariable
  - QEImage, [139](#)
- regionOfInterest4HVariable
  - QEImage, [139](#)
- regionOfInterest4WVariable
  - QEImage, [139](#)
- regionOfInterest4XVariable
  - QEImage, [139](#)
- regionOfInterest4YVariable
  - QEImage, [139](#)
- released
  - QECheckBox, [87](#)
  - QEPushButton, [193](#)
  - QERadioButton, [212](#)
- releaseText
  - QECheckBox, [93](#)
  - QEPushButton, [198](#)
  - QERadioButton, [217](#)
- RESIZE\_OPTION\_FIT
  - QEImage, [133](#)
- RESIZE\_OPTION\_ZOOM
  - QEImage, [133](#)
- resizeOption
  - QEImage, [139](#)
- ResizeOptions
  - QEImage, [133](#)
- resizeOptions
  - QEImage, [133](#)
- restore
  - SaveRestoreSignal, [269](#)
- restoreConfiguration
  - QEPvProperties, [203](#)
  - QEStripChart, [253](#)
  - QEWidget, [266](#)
- RGB\_888
  - QEImage, [132](#)
- Right\_To\_Left
  - QEAnalogIndicator, [66](#)
- ROIInfo, [268](#)
- Rotate180
  - QEImage, [134](#)
- Rotate90Left
  - QEImage, [134](#)
- Rotate90Right
  - QEImage, [134](#)
- rotation
  - QEImage, [139](#)
- ROTATION\_0
  - QEImage, [133](#)
- ROTATION\_180
  - QEImage, [133](#)
- ROTATION\_90\_LEFT
  - QEImage, [133](#)
- ROTATION\_90\_RIGHT
  - QEImage, [133](#)
- RotationOptions
  - QEImage, [133](#)
- rotationOptions
  - QEImage, [133](#)
- save
  - SaveRestoreSignal, [269](#)
- saveConfiguration
  - QEPvProperties, [203](#)
  - QEStripChart, [253](#)

- QEWidgit, 266
- SaveRestoreSignal, 269
  - restore, 269
  - save, 269
- saveRestoreSlot, 269
- Scale
  - QEAAnalogIndicator, 66
- scale2
  - QEShape, 235
- scale3
  - QEShape, 235
- scale4
  - QEShape, 235
- scale5
  - QEShape, 235
- scale6
  - QEShape, 235
- scaleBy
  - QEPvProperties, 204
  - QEWidgit, 266
- Scientific
  - QEAAnalogProgressBar, 71
  - QECheckBox, 86
  - QELabel, 148
  - QELineEdit, 157
  - QEPushButton, 191
  - QERadioButton, 210
- Scientist
  - QEAAnalogProgressBar, 72
  - QEBitStatus, 78
  - QECheckBox, 86
  - QEComboBox, 98
  - QEFram, 111
  - QEGenericEdit, 117
  - QEGroupBox, 122
  - QEImage, 134
  - QELabel, 149
  - QEPeriodic, 175
  - QEPlot, 184
  - QEPushButton, 192
  - QEPvProperties, 203
  - QERadioButton, 211
  - QEShape, 230
  - QESlider, 240
  - QESpinBox, 245
- selectMenu, 270
- selectOptions
  - QEImage, 134
- setConfirmWrite
  - QEGenericEdit, 118
- setLocalEnumeration
  - QELocalEnumeration, 163
- setMessageSourceId
  - QEWidgit, 267
- setSubscribe
  - QEGenericEdit, 118
- setupContextMenu
  - QEWidgit, 267
- setVariableNameAndSubstitutions
  - QEBitStatus, 78
  - QEFram, 109
  - QEWidgit, 267
- setWriteOnEnter
  - QEGenericEdit, 118
- setWriteOnFinish
  - QEGenericEdit, 118
- setWriteOnLoseFocus
  - QEGenericEdit, 118
- shapeOptions
  - QEShape, 229
- showScale
  - QEAAnalogIndicator, 67
- showText
  - QEAAnalogIndicator, 68
- showTime
  - QEImage, 140
- SO\_AREA4
  - QEImage, 134
- SO\_BEAM
  - QEImage, 134
- SO\_HSLICE
  - QEImage, 134
- SO\_NONE
  - QEImage, 134
- SO\_PANNING
  - QEImage, 134
- SO\_PROFILE
  - QEImage, 134
- SO\_TARGET
  - QEImage, 134
- SO\_VSLICE
  - QEImage, 134
- spanAngle
  - QEAAnalogIndicator, 68
- standardProperties, 270
- State
  - QECheckBox, 86
  - QEPushButton, 191
  - QERadioButton, 211
- StateMachineTemplate, 272

- subscribe
  - QCheckBox, [93](#)
  - QComboBox, [99](#)
  - QGenericEdit, [119](#)
  - QPeriodic, [176](#)
  - QPushButton, [198](#)
  - QRadioButton, [217](#)
  - QSlider, [241](#)
  - QSpinBox, [245](#)
- targetColor
  - QImage, [140](#)
- targetTriggerVariable
  - QImage, [140](#)
- targetXVariable
  - QImage, [140](#)
- targetYVariable
  - QImage, [140](#)
- Text
  - QCheckBox, [86](#)
  - QLabel, [149](#)
  - QPushButton, [191](#)
  - QRadioButton, [211](#)
- TextAndIcon
  - QCheckBox, [86](#)
  - QPushButton, [191](#)
  - QRadioButton, [211](#)
- textSubstitutions
  - QSubstitutedLabel, [260](#)
- textToDouble
  - QLocalEnumeration, [164](#)
- textToInt
  - QLocalEnumeration, [164](#)
- textToValue
  - QLocalEnumeration, [164](#)
- Time
  - QAnalogProgressBar, [71](#)
  - QCheckBox, [86](#)
  - QLabel, [148](#)
  - QLineEdit, [157](#)
  - QPushButton, [191](#)
  - QRadioButton, [210](#)
- timeColor
  - QImage, [140](#)
- Top\_To\_Bottom
  - QAnalogIndicator, [66](#)
- trace, [273](#)
- TrackRange, [273](#)
- trailingZeros
  - QAnalogProgressBar, [74](#)
  - QCheckBox, [93](#)
  - QLabel, [153](#)
  - QLineEdit, [159](#)
  - QPushButton, [198](#)
  - QRadioButton, [218](#)
- UnsignedInteger
  - QAnalogProgressBar, [71](#)
  - QCheckBox, [86](#)
  - QLabel, [148](#)
  - QLineEdit, [157](#)
  - QPushButton, [191](#)
  - QRadioButton, [210](#)
- UPDATE\_PIXMAP
  - QLabel, [149](#)
- UPDATE\_TEXT
  - QLabel, [149](#)
- updateOption
  - QCheckBox, [93](#)
  - QLabel, [153](#)
  - QPushButton, [199](#)
  - QRadioButton, [218](#)
- UpdateOptions
  - QCheckBox, [86](#)
  - QLabel, [148](#)
  - QPushButton, [191](#)
  - QRadioButton, [210](#)
- updateOptions
  - QLabel, [149](#)
- useDbDisplayLimits
  - QAnalogProgressBar, [75](#)
- useDbEnumerations
  - QComboBox, [98](#)
- useDbPrecision
  - QAnalogProgressBar, [75](#)
  - QCheckBox, [93](#)
  - QLabel, [153](#)
  - QLineEdit, [159](#)
  - QPushButton, [199](#)
  - QRadioButton, [218](#)
- User
  - QAnalogProgressBar, [72](#)
  - QBitStatus, [78](#)
  - QCheckBox, [86](#)
  - QComboBox, [98](#)
  - QFrame, [111](#)
  - QGenericEdit, [117](#)
  - QGroupBox, [122](#)
  - QImage, [134](#)
  - QLabel, [149](#)

- QEPeiodic, 175
- QEPlot, 184
- QEPushButton, 192
- QEPvProperties, 203
- QERadioButton, 211
- QEShape, 230
- QESlider, 240
- QESpinBox, 245
- userInfoStruct, 274
- USERLEVEL\_ENGINEER
  - userLevelTypes, 275
- USERLEVEL\_SCIENTIST
  - userLevelTypes, 275
- USERLEVEL\_USER
  - userLevelTypes, 275
- userLevelEnabled
  - QEAnalogProgressBar, 75
  - QEBitStatus, 79
  - QECheckBox, 93
  - QEComboBox, 99
  - QEFrame, 111
  - QEGenericEdit, 119
  - QEGroupBox, 123
  - QEImage, 140
  - QELabel, 153
  - QEPeiodic, 177
  - QEPlot, 185
  - QEPushButton, 199
  - QEPvProperties, 204
  - QERadioButton, 218
  - QEShape, 235
  - QESlider, 241
  - QESpinBox, 245
- userLevelEngineerStyle
  - QEAnalogProgressBar, 75
  - QEBitStatus, 79
  - QECheckBox, 94
  - QEComboBox, 100
  - QEFrame, 111
  - QEGenericEdit, 119
  - QEGroupBox, 123
  - QEImage, 140
  - QELabel, 153
  - QEPeiodic, 177
  - QEPlot, 185
  - QEPushButton, 199
  - QEPvProperties, 205
  - QERadioButton, 218
  - QEShape, 236
  - QESlider, 241
- QESpinBox, 246
- UserLevels
  - QEAnalogProgressBar, 71
  - QEBitStatus, 78
  - QECheckBox, 86
  - QEComboBox, 98
  - QEFrame, 111
  - QEGenericEdit, 117
  - QEGroupBox, 122
  - QEImage, 134
  - QELabel, 149
  - QEPeiodic, 175
  - QEPlot, 184
  - QEPushButton, 191
  - QEPvProperties, 203
  - QERadioButton, 211
  - QEShape, 229
  - QESlider, 240
  - QESpinBox, 244
- userLevels
  - userLevelTypes, 275
- userLevelScientistStyle
  - QEAnalogProgressBar, 75
  - QEBitStatus, 80
  - QECheckBox, 94
  - QEComboBox, 100
  - QEFrame, 112
  - QEGenericEdit, 120
  - QEGroupBox, 123
  - QEImage, 141
  - QELabel, 153
  - QEPeiodic, 177
  - QEPlot, 185
  - QEPushButton, 199
  - QEPvProperties, 205
  - QERadioButton, 218
  - QEShape, 236
  - QESlider, 241
  - QESpinBox, 246
- userLevelSignal, 274
- userLevelSlot, 275
- userLevelTypes, 275
  - USERLEVEL\_ENGINEER, 275
  - USERLEVEL\_SCIENTIST, 275
  - USERLEVEL\_USER, 275
  - userLevels, 275
- userLevelUserStyle
  - QEAnalogProgressBar, 75
  - QEBitStatus, 80
  - QECheckBox, 94

- QComboBox, [100](#)
- QFrame, [112](#)
- QGenericEdit, [120](#)
- QGroupBox, [124](#)
- QImage, [141](#)
- QLabel, [154](#)
- QPeriodic, [177](#)
- QPlot, [186](#)
- QPushButton, [199](#)
- QEPvProperties, [205](#)
- QERadioButton, [219](#)
- QEShape, [236](#)
- QESlider, [241](#)
- QESpinBox, [246](#)
- userLevelVisibility
  - QAnalogProgressBar, [76](#)
  - QBitStatus, [80](#)
  - QCheckBox, [94](#)
  - QComboBox, [100](#)
  - QFrame, [112](#)
  - QGenericEdit, [120](#)
  - QGroupBox, [124](#)
  - QImage, [141](#)
  - QLabel, [154](#)
  - QPeriodic, [177](#)
  - QPlot, [186](#)
  - QPushButton, [200](#)
  - QEPvProperties, [205](#)
  - QERadioButton, [219](#)
  - QEShape, [236](#)
  - QESlider, [242](#)
  - QESpinBox, [246](#)
- UserMessage, [275](#)
- UserMessageSignal, [278](#)
- UserMessageSlot, [279](#)
- value
  - QAnalogIndicator, [68](#)
- ValueScaling, [280](#)
- valueToText
  - QLocalEnumeration, [164](#)
- variable
  - QAnalogProgressBar, [76](#)
  - QBitStatus, [80](#)
  - QCheckBox, [94](#)
  - QComboBox, [100](#)
  - QGenericEdit, [120](#)
  - QLabel, [154](#)
  - QEPushButton, [200](#)
  - QEPvProperties, [205](#)
- QERadioButton, [219](#)
- QESlider, [242](#)
- QESpinBox, [246](#)
- variable1
  - QPlot, [186](#)
  - QEShape, [236](#)
- variable2
  - QPlot, [186](#)
  - QEShape, [237](#)
- variable3
  - QPlot, [186](#)
  - QEShape, [237](#)
- variable4
  - QPlot, [186](#)
  - QEShape, [237](#)
- variable5
  - QEShape, [237](#)
- variable6
  - QEShape, [237](#)
- variableAsToolTip
  - QAnalogProgressBar, [76](#)
  - QBitStatus, [80](#)
  - QCheckBox, [95](#)
  - QComboBox, [100](#)
  - QFrame, [112](#)
  - QGenericEdit, [120](#)
  - QGroupBox, [124](#)
  - QImage, [141](#)
  - QLabel, [154](#)
  - QPeriodic, [178](#)
  - QPlot, [187](#)
  - QEPushButton, [200](#)
  - QEPvProperties, [205](#)
  - QERadioButton, [219](#)
  - QEShape, [237](#)
  - QESlider, [242](#)
  - QESpinBox, [247](#)
- variableSubstitutions
  - QAnalogProgressBar, [76](#)
  - QBitStatus, [80](#)
  - QCheckBox, [95](#)
  - QComboBox, [101](#)
  - QGenericEdit, [120](#)
  - QImage, [141](#)
  - QLabel, [154](#)
  - QPeriodic, [178](#)
  - QPlot, [187](#)
  - QEPushButton, [200](#)
  - QEPvProperties, [206](#)
  - QERadioButton, [219](#)

- QEShape, [237](#)
- QESlider, [242](#)
- QESpinBox, [247](#)
- QEStriptChart, [253](#)
- verticalFlip
  - QEImage, [142](#)
- vertSliceColor
  - QEImage, [142](#)
- VideoWidget, [280](#)
- visible
  - QEAnalogProgressBar, [76](#)
  - QEBitStatus, [81](#)
  - QECheckBox, [95](#)
  - QEComboBox, [101](#)
  - QEFrame, [112](#)
  - QEGenericEdit, [121](#)
  - QEGroupBox, [124](#)
  - QEImage, [142](#)
  - QELabel, [154](#)
  - QEPeriodic, [178](#)
  - QEPlot, [187](#)
  - QEPushButton, [200](#)
  - QEPvProperties, [206](#)
  - QERadioButton, [219](#)
  - QEShape, [238](#)
  - QESlider, [242](#)
  - QESpinBox, [247](#)
- WidgetRef, [281](#)
- widthVariable
  - QEImage, [142](#)
- writeButtonVariable1
  - QEPeriodic, [178](#)
- writeButtonVariable2
  - QEPeriodic, [178](#)
- writeNow
  - QEWidget, [267](#)
- writeOnChange
  - QEComboBox, [98](#)
  - QESlider, [240](#)
- writeOnClick
  - QECheckBox, [95](#)
  - QEPushButton, [200](#)
  - QERadioButton, [219](#)
- writeOnEnter
  - QEGenericEdit, [121](#)
- writeOnFinish
  - QEGenericEdit, [121](#)
- writeOnLoseFocus
  - QEGenericEdit, [121](#)
- writeOnPress
  - QECheckBox, [95](#)
  - QEPushButton, [201](#)
  - QERadioButton, [220](#)
- writeOnRelease
  - QECheckBox, [95](#)
  - QEPushButton, [201](#)
  - QERadioButton, [220](#)
- Zoom
  - QEImage, [133](#)
- zoomMenu, [282](#)