

<Project Name>  
<Version Number> 0.1.0???

## Technical Documentation

### Gauge Widget Plugin API Details

# Table of Contents

Revision History.....	3
Introduction.....	3
Installation under Linux.....	4
Installation under Win32.....	4
Using The Widget.....	4
Signals.....	5
changed().....	5
Slots.....	5
void setPlateType(int).....	5
int plateType().....	5
void setArrowType(int).....	5
int arrowType().....	6
void setAntialias(bool).....	6
bool antialias().....	6
void setValue(int).....	6
void setValue(double).....	6
double value().....	7
int value_int().....	7
void setMaximum(double).....	7
double maximum().....	7
void setMinimum(double).....	7
double minimum().....	7
void setLabel(QString).....	7
QString label().....	7
void setTics(int).....	7
int tics().....	7
void setSubTics(int).....	8
int subTics().....	8
void setStartAngle(double).....	8
double startAngle().....	8
void setEndAngle(double).....	8
double endAngle().....	8
void setChevronWidth(double).....	8
double chevronWidth().....	8
void setQuality(int).....	8
int quality().....	9
void setBorder(int).....	9
int border().....	9

## Revision History

Date	Revision	Detail
9 <sup>th</sup> Dec '07	0	

## Introduction

This widget is intended to provide an easy way to draw custom gauges. The widget is extendible and highly configurable, with the option to draw simple, or more complex gauges as required. The user has a considerable amount of control over the design of a base gauge, as well as being able to choose the design (plate) and arrow individually.

## Installation under Linux

Here we assume that the gauge widget is unzipped to ~/gauge and the project you wish to include the gauge in is ~/project. The following commands will install the gauge widget:

```
cd ~/gauge
make
su root
make install
cd /usr/lib
ln -s /usr/lib/qt4/plugins/designer/libgauge.so
logout

cd ~/project
mkdir include
cd include
ln -s ~/gauge/gauge.h
```

If you intend to develop the plugin, it is worth giving yourself write permission to /usr/lib/qt4/plugins/designer, this will save a lot of time when re-building the library! This process only needs to be done once. Any update to the widget will be available to the project due to the links.

## Installation under Win32

Here we assume that the gauge widget is unzipped to D:\gauge and the project you wish to include the gauge in is D:\project. The following commands will install the gauge widget:

```
cd D:\gauge
make install
copy release\gauge.dll C:\Qt\4.3.3\lib\gauge.dll

mkdir D:\project\include
copy D:\gauge\gauge.h D:\project\include\gauge.h
```

This process must be repeated every time the widget is updated. It is advisable to write a batch script to automate this process if you intend to develop the widget under windows.

## Using The Widget

Once you have successfully installed the widget, open QTDesigner and you will see a new widget called “Gauge” in the “Display Widgets” section of QTDesigner. You can now just drag and drop the widget as you would for any other. However, you must ensure that your project file has the following line in the global section.

```
LIBS += -lgauge
```

**Signals**

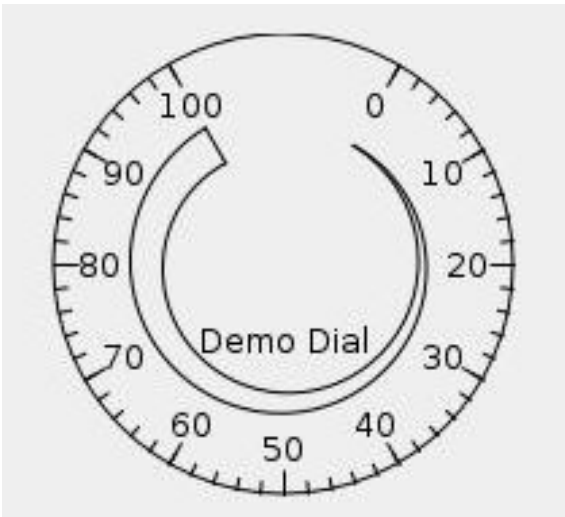
**changed()**

This signal tells the rest of the form that the widget has changed. This may be useful in certain circumstances.

**Slots**

**void setPlateType(int)**

This slot allows the user to choose which gauge style to use. The designation “plate” comes from the dial-plate used on “proper” gauges.

Plate	Illustration
0	


Default: 0

**int plateType()**

Returns the current plate.

**void setArrowType(int)**

This slot allows the user to choose which arrow (or “needle”) to use on the gauge.

Arrow	Illustration
0	

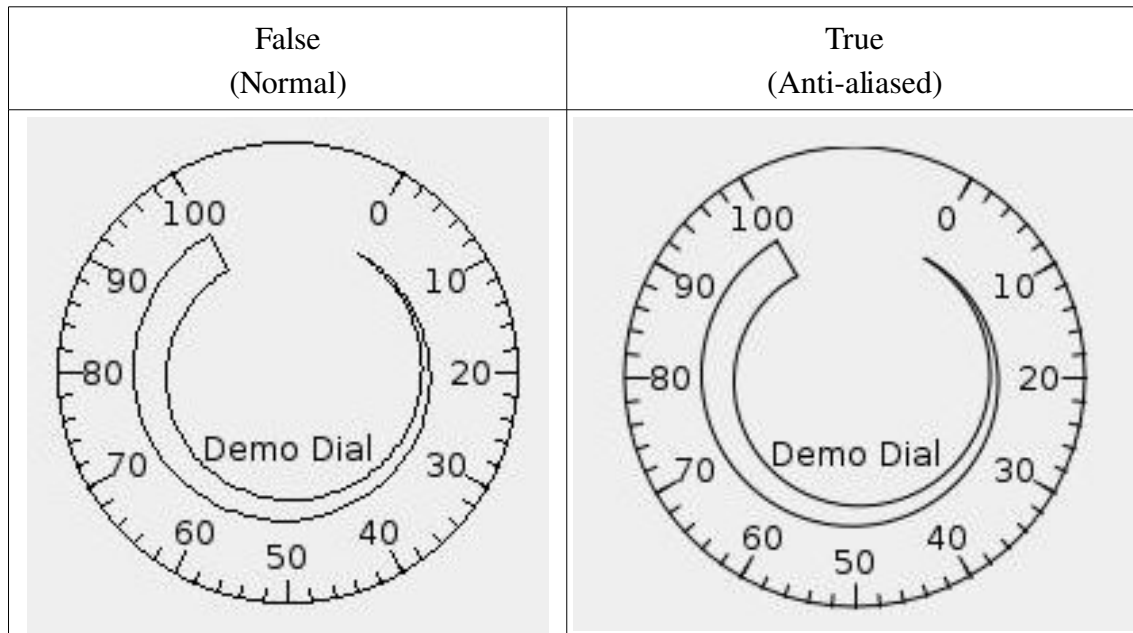
Default: 0

## **int arrowType()**

Returns the current arrow.

## **void setAntialias(bool)**

Allows the user to anti-alias the gauge. This produces “smoother” gauges, but requires a little more CPU power. It is recommended that anti-aliasing is used.



Default: True

## **bool antialias()**

Returns true if anti-aliasing is used.

## **void setValue(int)**

Sets the gauge to an integer value. This is provided to accept signals from other widgets. This is internally limited to be within the limits of the scale.

Default: 13

## **void setValue(double)**

Sets the gauge to a double-precision value. This is the normal way to set the gauge value. This is internally limited to be within the limits of the scale.

Default: 13

## **double value()**

Returns the current gauge value as a double-precision value.

## **int value\_int()**

Returns the gauge value to the nearest integer.

## **void setMaximum(double)**

Sets the maximum value that can be plotted on the scale.

Default: 100

## **double maximum()**

Returns the maximum limit of the scale.

## **void setMinimum(double)**

Sets the minimum value that can be plotted on the scale.

Default: 0

## **double minimum()**

Returns the minimum limit of the scale

## **void setLabel(QString)**

Sets the gauge label.

Default: "Demo Dial"

## **QString label()**

Returns the gauge label

## **void setTics(int)**

Sets the number of major tick-marks around the dial.

Default: 11

## **int tics()**

Returns the number of major tick-marks

## **void setSubTics(int)**

Sets the number of minor tick marks which make up each major tick. For example, to space minor tick-marks at 0.2, between major tick-marks 0 and 1, use setSubTics(5).

Default: 5

## **int subTics()**

Returns the number of minor tick-marks that make up one major tick-mark

## **void setStartAngle(double)**

Sets the start angle for the scale. This may be greater than the end angle or greater than 360 degrees.

Default: 30

## **double startAngle()**

Returns the start angle for the scale.

## **void setEndAngle(double)**

Sets the end angle for the scale. This may be less than the start angle, or greater than 360 degrees. Using start angle and end angle together allows for all gauge orientations

Default: 330

## **double endAngle()**

Returns the end angle for the scale.

## **void setChevronWidth(double)**

Sets the current maximum width of the chevron for plate 0.

Default: 15

## **double chevronWidth()**

Returns the current maximum width of the chevron for plate 0.

## **void setQuality(int)**

Sets the number of points used to draw the chevron on plate 0. This should be increased as the gauge is made bigger, and may be reduced as it is made smaller.



Default: 120

### **int quality()**

Returns the current number of points used to draw the chevron on plate 0.

### **void setBorder(int)**

Reduces the radius of the gauge by the desired number of pixels, whilst keeping the widget-size the same to make using gauges in grid layouts easier.

Default: 0

### **int border()**

Returns the current border.