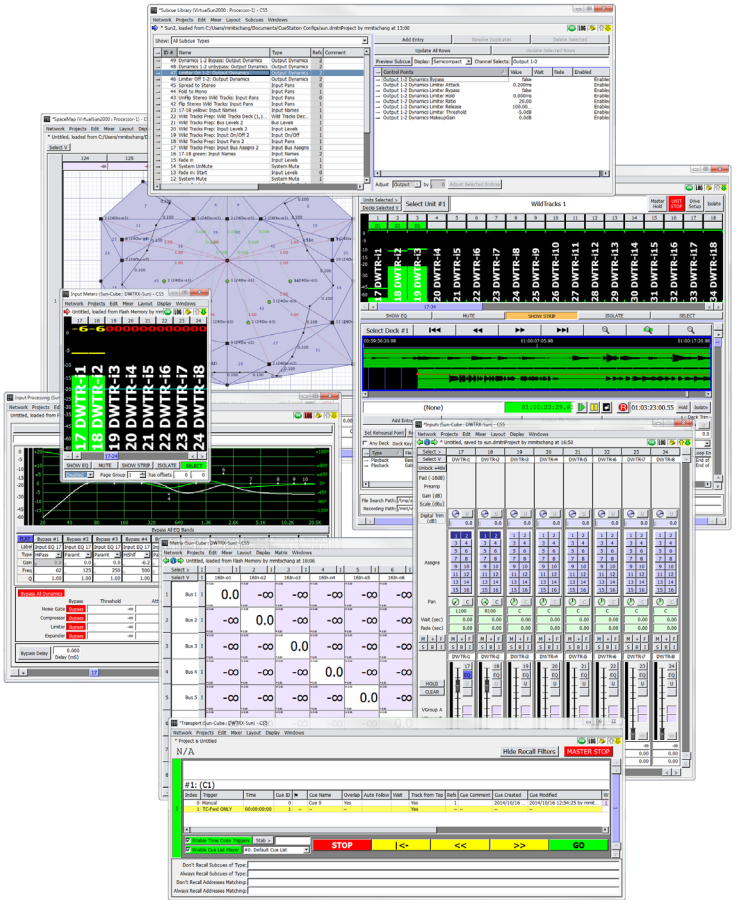
CueStation Client-Side Python

thinking sound Meyer Sound



python™

Keep these important operating instructions.
Check www.meyersound.com for updates.

CueStation Client-Side Python User Guide, PN 05.176.131.02 A

CONTENTS

CHAPTER 1: INTRODUCTION

HOW TO USE THIS MANUAL

As you read this user guide, you will encounter the following icons for notes, tips, and cautions:

NOTE: A note identifies an important or useful piece of information relating to the topic under discussion.

TIP: A tip offers a helpful tip relevant to the topic at hand.

CAUTION: A caution gives notice that an action may have serious consequences and could cause harm to equipment or personnel, or could cause delays or other problems.

Information and specifications are subject to change. Updates and supplementary information are available at www.meyersound.com.

Meyer Sound Technical Support is available at:

- **Tel:** +1 510 486.1166
- **Tel:** +1 510 486.0657 (after hours support)
- **Web:** www.meyersound.com/support
- **Email:** techsupport@meyersound.com

OVERVIEW

This document describes how Python® scripting can be used to construct custom CueStation GUI windows in CueStation 5.5.2 or higher.

Note that this document assumes at least a basic level of understanding of the Python scripting language. If you are unfamiliar with Python, it's recommended that you spend some time learning basic Python before continuing with this document.

There are plenty of good Python tutorials and documentation available online and offline, including these:

- Learning Python
- Byte of Python
- Begin Python

PYTHON AND CUESTATION 5

D-Mitri and CueStation have long supported the running of Python scripts on the D-Mitri servers as a way to automate various tasks, but new for CueStation 5.5.2 is the ability to run Python scripts inside a CueStation window on a client computer. This is useful because a Python script running there can direct CueStation to populate the window with various GUI widgets, and (optionally) interact with the window while it is running in order to implement custom GUI behaviors. This gives the CueStation's GUI much more flexibility than it previously had, because the user is no longer limited to the built-in GUI designs that come with CueStation -- you can now make your own CueStation windows to suit your tastes.

CHAPTER 2: RUNNING A CLIENT-SIDE PYTHON SCRIPT

SELECTING CLIENT-SIDE PYTHON SCRIPTS

In order to run a client-side Python script, the user must first make sure that the Python script is listed in the Support Files window.

To place a Python script in the Support Files window, do one of the following:

- Choose Windows > Support Files, then drag your Python script file from your desktop into the Support Files window.
- Choose Windows > Support Files, then choose Files > Import Files and choose a Python script file from the browser.

RUNNING CLIENT-SIDE PYTHON SCRIPTS

When the client-side Python script is running, a message will be displayed in the Custom window's title bar (such as "test_window.py is running"). The Python script will continue to run until it is instructed to exit, or until the host window is closed or disconnected from the server. If any of the files that the script depends on in the Support File window are modified, the script will automatically be stopped and restarted using the updated file versions -- so if you are developing a client-side Python script and wish to audition your changes, all you need to do is re-import the *.py* file, and any Custom windows using that file will update themselves automatically.

NOTE: Any errors or messages printed by the Python script will be listed in the Log window, which is helpful for debugging. Alternatively, if you run CueStation from the command line you will also see the Python script's output printed to the terminal window.

There are multiple ways to run client-side Python scripts in CueStation. The following methods integrate Python scripts into your CueStation 5 project file and allow for multiple ways of running scripts.

Running Client-Side Scripts from the Support Files Window

To run a client-side Python script from the Support Files window:

1. Choose Windows > Support Files.
2. Right click the name of the your script, and choose Run Selected Script as CueStation Window.

Running Client-Side Scripts from the Windows Menu

To run a client-side Python script from the Windows menu:

1. Open the Windows menu, and select the window type that corresponds to your Python script at the bottom of the menu.

TIP: Only scripts with *_window* appended to the file names are available from the Windows menu. For example, adding *MyScript_window.py* to Support Files results in the entry Windows > MyScript. Conversely, adding *MyScript.py* to Support Files will not result in a Windows menu entry.

Running Client-Side Scripts from a Saved D-Mitri Layout

To run a client-side Python script from the Windows menu:

1. Open the Windows Menu, and select the window type that corresponds to your Python script at the bottom of the menu.
2. Choose one of the following to save the current layout:

- Layout > Save Layout to save the current layout.
- Layout > Save Layout As to save the current layout as a file.
- Layout > Save Layout As Support File to save the current layout to the Support Files window.
- Layout > Save Layout As Default to save the current layout as the default layout for the client computer.

3. Open the layout with your saved client-side Python script, do one of the following:

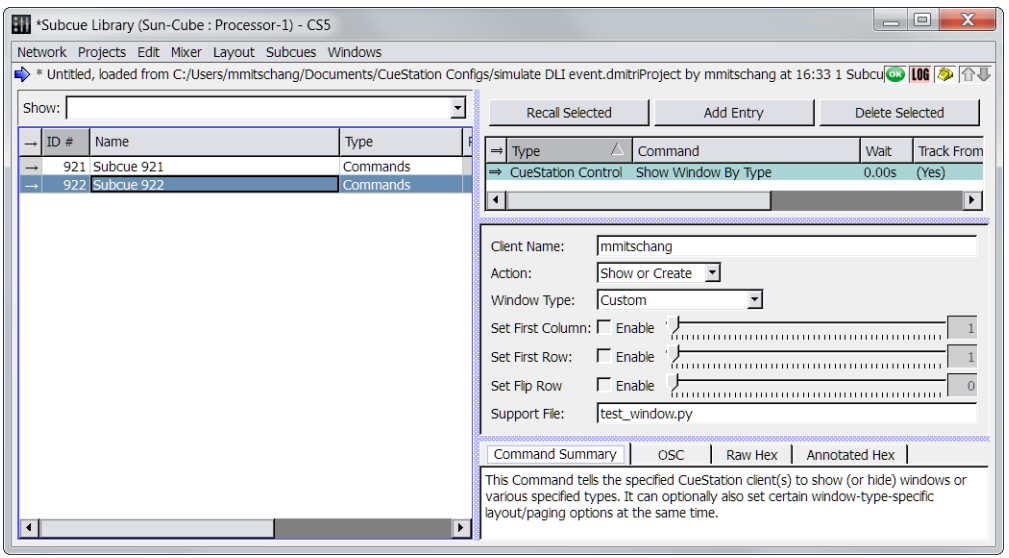
- Choose Layout > Open Layout and choose a .dmitriLayout file from the browser.

- Choose Layout > Open Layout from Support File and choose a .dmitriLayout file from the list of Support Files.

Running Client-Side Scripts with a CueStation Control Subcue Entry

To run a client-side Python script using a Command subcue of type CueStation Control:

1. Choose Windows > Support Files. Drag the script file into the Support Files window.
2. In CueStation's Subcue Library window, choose Subcues > New Subcue > Commands to create a new Command subcue.
3. Click Add Entry to create a new subcue entry. Set the Type to CueStation Control, then set the Command to Show Window by Type.
4. Set the Action to Show or Create.
5. Set the Window Type to Custom. Enter the filename of the script that you added to Support Files.



INCLUDED PYTHON SCRIPTS

CueStation 5 and the D-Mitri firmware comes with a number of client-side Python scripts included. These scripts can always be run using any of the techniques described later in this chapter, since they are in the Python interpreter's default path. These files are located in the following directories:

- Windows 32-bit operating systems:
  C:\Program Files\Meyer Sound\CueStation 5\templates_guiscripts
- Windows 64-bit operating systems:
  C:\Program Files (x86)\Meyer Sound\CueStation 5\templates_guiscripts
- Mac OS X operating systems:
  Right-click the VirtualD-Mitri icon in the Finder and choose Show Package Contents, then navigate to *Contents/Resources/templates/_guiscripts*

These scripts can be used as-is, and they also serve as a set of examples showing how to implement various useful tasks as Python scripts under D-Mitri. A quick way to get started is to make a copy of a script file, modify the copy to suit your needs, then import the copy into the Support Files window.

HOW A CLIENT-SIDE PYTHON SCRIPT WORKS

CueStation runs a client-side Python script on the client machine in a way that is very similar to how D-Mitri runs a server-side Python script. Once CueStation has downloaded the necessary Support Files from the server, it launches a child process that executes the Python interpreter on the downloaded *.py* file. This *.py* file can in principle be any Python program you like, but in practice you will almost always want to make use of the new GUIClient Python class (included in the templates/_meyer folder). The GUIClient class will do all of the necessary setup and communication to allow your Python script to communicate with the CueStation window and the D-Mitri servers.

TIP: The GUIClient class is an extension of the BasicClient class that is used for server-side Python scripting, which means that anything you could previously do in a server-side Python script you can now also do inside a client-side script.

The following is a simple client-side Python script:

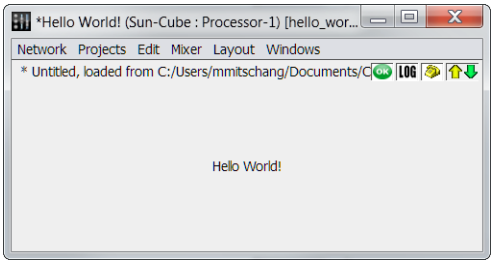
```
from dmitri_script import *

class HelloWorldGUIClient(GUIClient):
   def SetupGUI(self, layoutState):
      self.SetWindowTitle("Hello World!")
      self.AddWidget(name="Hello", text="Hello World!", type="label", alignment="center")

if __name__ == "__main__":
   HelloWorldGUIClient().RunGUIEventLoop()
```

NOTE: The above script can be pasted into a text editor, saved as hello_world_window.py or similar, and run from within the Support Files window. Alternatively, this file can be imported from the templates/_guiscripts folder, as described in "Selecting Client-Side Python Scripts" on page 7.

This script creates a simple CueStation window with the words “Hello World!” in the center of the window and in the window’s title bar:



Inside the “Hello World!” Script

```
from dmitri_script import *
```

The above line instructs the Python interpreter to use all of the standard Python support code that comes with CueStation. While it is possible to manually import each Python support file individually (as in “`from gui_client import *`”, “`from control_point_address import *`”, and so on), the templates/_meyer/dmitri_script.py file will then import everything else automatically.

```
class HelloWorldGUIClient(GUIClient):
```

The above line instructs the Python interpreter to define a subclass of the `GUIClient` class, and this subclass is to be called `HelloWorldGUIClient`. The `HelloWorldGUIClient` class will therefore have all the capabilities of the standard `GUIClient` class (defined in templates/_meyer/gui_client.py), plus additional functionality (specifically, the “Hello World!” label) that will be added with subsequent commands.

```
def SetupGUI(self, layoutState):
   self.SetWindowTitle(“Hello World!”)
   self.AddWidget(name=“hello”, type=“label”, text=“Hello World!”, alignment=“center”)
```

The above lines define the `SetupGUI(self, layoutState)` method of the class. The event loop of the `GUIClient` class will call this method once all of the groundwork has been prepared and the CueStation window is ready for use. The implementation of this method sets the window title to “Hello World!” and then adds a single GUI widget to the center of the window. The widget is named “hello,” separate from the text it is to display (“Hello World!”). Naming a widget is optional, but doing so makes it possible for the user to refer to the widget again later on in the script, if desired. Widget names all share a single namespace, so no two widgets can share the same name. The `layoutState` argument is a Message that may contain a persistent state that was saved by a previous incarnation of this window via the `SaveLayoutState(self, msg, optPath)` method. It can optionally be used to keep state data for the script that goes with the window’s .dmitriLayout file, etc.

```
if __name__ == "__main__":
   HelloWorldGUIClient().RunGUIEventLoop()
```

The above is necessary boilerplate code that will be found at the bottom of almost every client-side Python script. The second line creates a `HelloWorldGUIClient` object, which is different from the previous code that defined the `HelloWorldGUIClient` class of objects. Once the `HelloWorldGUIClient` object is created, it immediately calls the `RunGUIEventLoop(self)` method on that object. The `RunGUIEventLoop(self)` method will not return until the script is ready to exit, because it is running an event loop that will handle all of the script’s needs. In particular, it will call the `SetupGUI(self, layoutState)` method at the appropriate time, and other methods such as `GUIControlChanged(self, address, value)` at the appropriate times as well.

CHAPTER 3: PYTHON AND CUESTATION 5 GUI

ADDING WIDGETS TO THE GUI

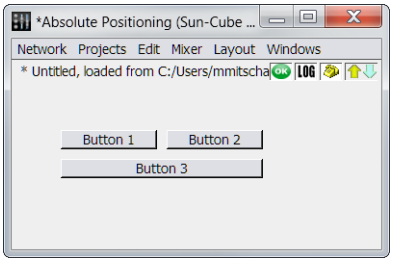
The first thing most client-side scripts aim to do is add widgets to the GUI. CueStation currently supports dozens of different widget types. As seen in the previous section, the way to add a widget to the window is to call `self.AddWidget(self, **args)` with a type="something" keyword argument indicating what type of widget should be created. The various widget-type strings that are currently supported are shown in the first column of Appendix A, "Index Tables". In addition to specifying the type of widget to create, other widget properties can be created to control various attributes of the widget. In the example in the previous section, "text" was one such property, and it was set to "Hello World!". There are many other properties available to be set, and different widget types can have different properties applied to them, as described in the per-widget class pages in Appendix B, "Container Types" and Appendix C, "Control Types". The second thing that must be specified after deciding what widget to add, is where to add it. There are two basic approaches to specifying widget positioning. The first approach is to give explicit pixel coordinates, as shown in this example script:

```
from dmitri_script import *

class AbsolutePositioningGUIClient(GUIClient):
   def SetupGUI(self, layoutState):
      self.SetWindowTitle("Absolute Positioning")
      self.AddWidget(type="pushbutton", text="Button 1", geometry=(50,50,100,20))
      self.AddWidget(type="pushbutton", text="Button 2", geometry=(160,50,100,20))
      self.AddWidget(type="pushbutton", text="Button 3", geometry=(50,80,210,20))

if __name__ == "__main__":
   AbsolutePositioningGUIClient().RunGUIEventLoop()
```

This script specifies pixel coordinates for each of the three buttons we add to the GUI (left,top,width,height). This results in the following GUI:



This method of widget layout is conceptually simple, and offers complete control over the static positioning of widgets. However, this method has the following restrictions:

- The widgets will not resize or reposition when the window is resized
- The widgets are not optimized for different font sizes
- All coordinates (left,top,width,height) must be determined manually
- As a result of each module's hard-coded pixel coordinates, it will be difficult to break up a large GUI into independent, reusable modules

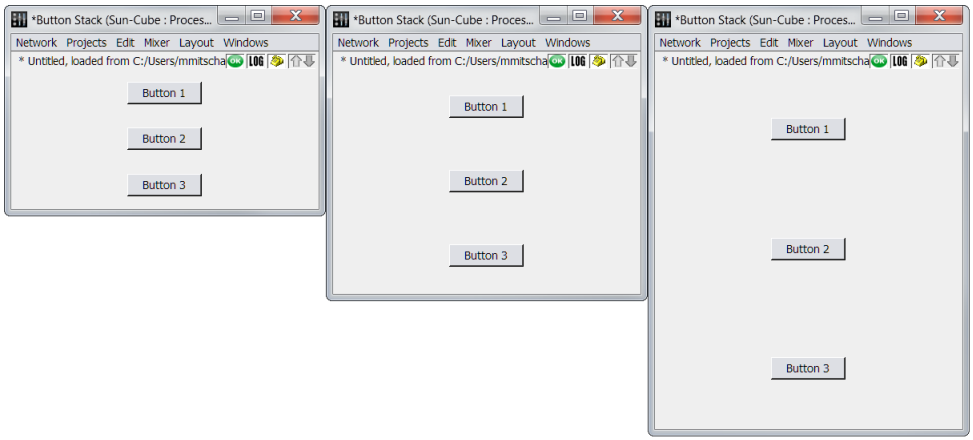
The alternative to manual widget layout is managed layout, in which the windowing system will automatically calculate widget positions based on the current window size and some rules supplied by the user. For example, the following program lays out some buttons in a vertical stack that resizes with the window:

```
from dmitri_script import *

class ButtonStackGUIClient(GUIClient):
   def SetupGUI(self, layoutState):
      self.SetWindowTitle("Button Stack")
      with self.AddContainer(layout="vertical"):
         self.AddWidget(type="pushbutton", text="Button 1")
         self.AddWidget(type="pushbutton", text="Button 2")
         self.AddWidget(type="pushbutton", text="Button 3")

if __name__ == "__main__":
   ButtonStackGUIClient().RunGUIEventLoop()
```

This will result in the following layout changes when the window is resized:



In the above example, the button widgets would become the children of a “container widget”:

```
with self.AddContainer(layout="vertical"):
```

In the `AddContainer()` call, we specified that we wanted a “vertical” layout. Other supported layouts are the “horizontal” layout and the “grid” layout. Horizontal is very similar to the vertical layout, except that each child widget is added to the right of its previous sibling widget, rather than below it. In the grid layout, child objects are laid out according to the cells of a grid.

Each of these three layout types allows for a couple of extra arguments in the layout argument’s string. The two optional extra arguments can be used to indicate the layout manager’s desired “margin” and “spacing” respectively. For example, `layout="vertical,20,5"` would tell the container to lay out its child widgets vertically, with 20 pixels of margin around the edges of the container, and a minimum of 5 pixels of spacing between adjacent child widgets. If these extra arguments are not specified, reasonable default values will be used. This layout will not by itself allow you to lay out widgets in the non-trivial patterns used in most GUIs. The way to get more complex widget layouts is to “nest” multiple containers together into a hierarchy.

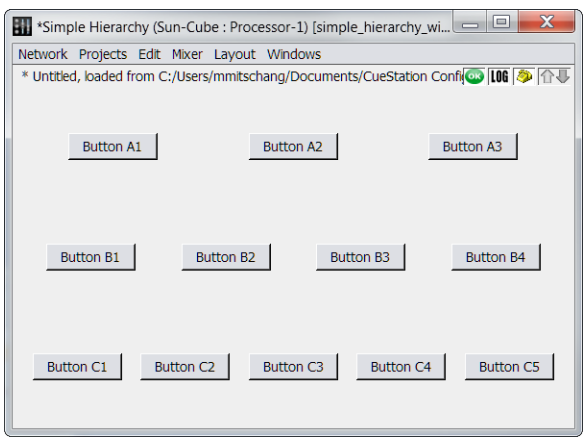
The following is an example script that creates three rows of containers, and arranges different numbers of buttons into each row:

```
from dmitri_script import *

class SimpleHierarchy(GUIClient):
   def SetupGUI(self, layoutState):
      self.SetWindowTitle("Simple Hierarchy")
      with self.AddContainer(layout="vertical,10"):
         with self.AddContainer(layout="horizontal"):
            self.AddWidget(type="pushbutton", text="Button A1")
            self.AddWidget(type="pushbutton", text="Button A2")
            self.AddWidget(type="pushbutton", text="Button A3")
         with self.AddContainer(layout="horizontal"):
            self.AddWidget(type="pushbutton", text="Button B1")
            self.AddWidget(type="pushbutton", text="Button B2')
            self.AddWidget(type="pushbutton", text="Button B3")
            self.AddWidget(type="pushbutton", text="Button B4')
         with self.AddContainer(layout="horizontal"):
            self.AddWidget(type="pushbutton", text="Button C1")
            self.AddWidget(type="pushbutton", text="Button C2")
            self.AddWidget(type="pushbutton", text="Button C3")
            self.AddWidget(type="pushbutton", text="Button C4")
            self.AddWidget(type="pushbutton", text="Button C5")

if __name__ == "__main__":
   SimpleHierarchy().RunGUIEventLoop()
```

This will result in the following window:

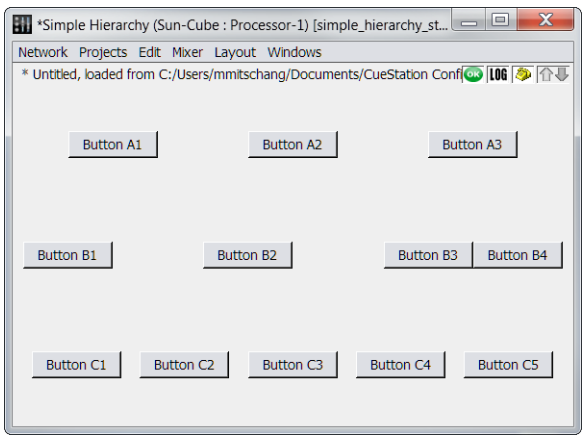


Containers can be nested as deeply as needed. For the sake of readability, it is best to break up the code into different methods if the nesting starts to get too deep (see the file /templates/_guiscripts/test_window.py for an example of that technique).

All examples so far have given all widgets an equal amount of space in their container. In order to dedicate more space to certain buttons, a “stretch factor” can be used to give priority over the other widgets. For example, the “Button B2” line of the above example can be modified:

```
self.AddWidget(type="pushbutton", text="Button B2", stretch=1)
```

The GUI shows more space around Button B2 once the modified script is imported, as illustrated in the following image:



It is possible to add a fixed number of extra pixels of space between two widgets:

```
[...]
     self.AddWidget(type="pushbutton", text="Button C2")
     self.AddWidget(type="spacing", spacing=50)
     self.AddWidget(type="pushbutton", text="Button C3")
[...]
```

It is also possible to add a “stretchy spring” between two widgets that pushes them apart:

```
    [...]
         self.AddWidget(type=“pushbutton”, text=“Button C2”)
         self.AddWidget(type=“stretch”)
         self.AddWidget(type=“pushbutton”, text=“Button C3”)
    [...]
```

There are various other parameters that can be applied to vertical and horizontal layout containers, or to the widgets they hold, to get various effects. For more information on these parameters, see Appendix A, “Index Tables”, Appendix B, “Container Types”, and Appendix C, “Control Types.”

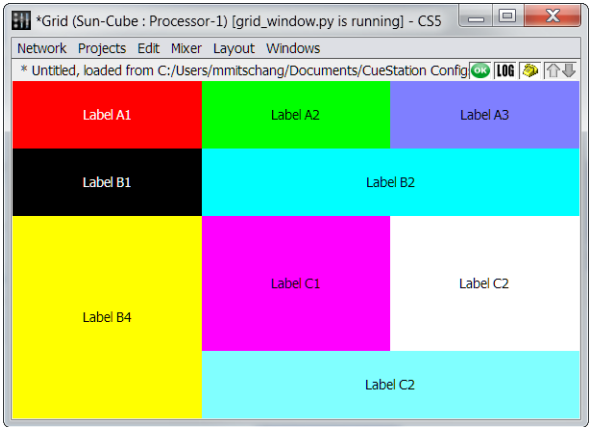
Below is a script that demonstrates the use of a grid layout. Widgets in a grid layout are not restricted to occupying just one square of the grid. Optional (minimum,maximum) values can be supplied for a widgets “row“ or “column” arguments instead of a single value, in order to get a layout where some widgets span multiple rows or columns of the layout manager’s grid:

```
    from dmitri_script import *

    class GridClient(GUIClient):
       def SetupGUI(self, layoutState):
          self.SetWindowTitle(“Grid”)
          with self.AddContainer(layout=“grid”):
             self.AddWidget(type=“label”, text=“Label A1”, alignment=“center”, bgColor=“red”,
row=0,col=0)
             self.AddWidget(type=“label”, text=“Label A2”, alignment=“center”,
bgColor=“green”, row=0, col=1)
             self.AddWidget(type=“label”, text=“Label A3”, alignment=“center”,
bgColor=“blue”, row=0, col=2)
             self.AddWidget(type=“label”, text=“Label B1”, alignment=“center”,
bgColor=“purple”, row=1, col=0)
             self.AddWidget(type=“label”, text=“Label B2”, alignment=“center”,
bgColor=“cyan”, row=1, col=(1,2))
             self.AddWidget(type=“label”, text=“Label B4”, alignment=“center”,
bgColor=“yellow”, row=(2,4), col=0)
             self.AddWidget(type=“label”, text=“Label C1”, alignment=“center”,
bgColor=“magenta”, row=(2,3), col=1)
             self.AddWidget(type=“label”, text=“Label C2”, alignment=“center”,
bgColor=“white”, row=(2,3), col=(2))
             self.AddWidget(type=“label”, text=“Label C2”, alignment=“center”,
bgColor=“128,255,255”, row=(4), col=(1,2))

    if __name__ == “__main__”:
       GridClient().RunGUIEventLoop()
```

This script results in the following image:



GETTING FEEDBACK FROM THE GUI

For many types of controls, notification is desired when the user manipulates the controls, so that an action can be made in response. The `GUIClient` class will notify when the user manipulates a control by calling the `GUIControlChanged()` method of the subclass, if it has one. The following is a simple implementation of the `GUIControlChanged()`:

```
    def GUIControlChanged(self, name, verb, val):
       print "GUI Control Changed! name=[%s] verb=[%s] val=[%s]" % (name, verb, val)
```

Implementing the `GUIControlChanged()` method as shown above will cause the script to print out the information it received to the Log, so notification can be seen. The following is a complete example:

```
    from dmitri_script import *

    class ControlChangedNotificationExampleClient(GUIClient):
       def __init__(self):
          self._counter = 0

       def SetupGUI(self, layoutState):
          with self.AddContainer(layout="vertical", sizePolicy="minimum"):
             self.SetWindowTitle("GUIControlChanged() Example")
             self.AddWidget(type="label", text="(Note: See Log window for notification event
info)", boxAlignment="center")
             self.AddWidget(type="pushbutton", name="theButton", text="Button")
             self.AddWidget(type="label", name="theLabel", text="Label",
boxAlignment="center", bgColor="yellow")
             self.AddWidget(type="label", name="theCountLabel", boxAlignment="center")
             self.AddWidget(type="slider", name="theSlider", fixedWidth=100,
boxAlignment="center")
             self.AddWidget(type="doublespinbox", name="theSpinBox", fixedWidth=100,
boxAlignment="center")
             self.AddWidget(type="checkbox", name="theCheckBox", text="Checkbox")

       def GUIControlChanged(self, name, verb, val):
          print "GUI Control Changed!  name=[%s] verb=[%s] val=[%s]" % (name, verb, val)
          if (name == "theButton"):
             if (verb == "pressed"):
             self.SetWidgetProperty("theLabel", text="Button Pressed!")
          elif (verb == "released"):
             self.SetWidgetProperty("theLabel", text="Button Released!")
          elif (verb == "clicked"):
               self._counter = self._counter+1
               self.SetWidgetProperties("theCountLabel", text="Click Count is now
%i!"%self._counter, bgColor="green")

    if __name__ == "__main__":
       ControlChangedNotificationExampleClient().RunGUIEventLoop()
```

The above script displays a few different widgets and reacts when some of them are pressed. It also prints to the log whenever a control-change notification is received.

SETTING PROPERTIES ON A WIDGET

Most of the properties that can be specified as arguments in a call to `AddWidget()` can also be specified later on to change an existing widget, by calling `SetWidgetProperties()`. For example, in the ControlChangedNotificationExampleClient script shown in the previous section, the `GUIControlChanged()` method calls `SetWidgetProperties()` on a label widget to update the text it displays, and turn its background color green:

```
    self.SetWidgetProperties("theCountLabel", text="Click Count is now %i!"%self._counter,
bgColor="green")
```

NOTE: `SetWidgetProperty()` is a synonym for `SetWidgetProperties()`. Both methods operate in exactly the same way.

`SetWidgetProperty()` and `SetWidgetProperties()` only operate on widgets that were given names. They cannot update anonymous widgets, since they have no way to specify them.

One interesting feature of the `SetWidgetProperty()` or `SetWidgetProperties()` method is that it supports wildcarding in the name argument. For example, the background color of all the named widgets in the above example will be set to red with the following single call:

```
    self.SetWidgetProperties("the*", bgColor="red")
```

The list of properties available for each of the various widget types can be found in the reference documentation (Appendix A, “Index Tables”, Appendix B, “Container Types”, and Appendix C, “Control Types”).

INVOKING METHODS ON A WIDGET

Some useful actions can be applied to a widget, which cannot be expressed as properties. For example, adding an item to a combo box or causing a button to click itself are momentary actions, not attributes. To invoke a method on a widget, the `InvokeWidgetMethod()` method can be used:

```
    self.InvokeWidgetMethod("theButton", "animateClick")
```

Various arguments can be specified for the methods, as necessary. For example, the following script is a method invocation that creates a combobox, and uses `InvokeWidgetMethod()` to add items to it. Since the "addItem" method of the combobox class requires a single string argument, a single string argument is passed after the name of the method to call.

```
self.AddWidget(name="theCombo", type="combobox")
self.InvokeWidgetMethod("theCombo", "addItem", "Item 1")
self.InvokeWidgetMethod("theCombo", "addItem", "Item 2")
self.InvokeWidgetMethod("theCombo", "addItem", "Item 3")
```

The same method can be invoked on multiple widgets by adding wildcarding to the first argument of the `InvokeWidgetMethod()` call. The list of methods available for each of the various widget types can be found in the reference documentation (Appendix A, "Index Tables", Appendix B, "Container Types", and Appendix C, "Control Types").

OTHER METHODS AVAILABLE IN THE GUIClient CLASS

There are various other methods available in the `GUIClient` class to accomplish various things. They are described briefly here.

TIP: View the file templates/_meyer/gui_client.py for more information about `GUIClient` methods.

Method Name	Description
RunGUIEventLoop(self)	This method runs the GUIClient event loop. It will not return until it is time for the script to exit. It should be called at the bottom of the script file, as shown in the example in section 3.
GetMixerConfiguration(self)	This method returns the current Mixer Configuration object. It can be used to find out, for example how many inputs/outputs/etc there are in the current system configuration, and their indices. See templates/_guiscripts/test_window.py for an example of this technique.
GetLocalUserName(self)	This method returns the name of the local window's CueStation client (such as what is specified via "Network -> Set Client Name" in the GUI)

Method Name	Description
GUICommandBatch(self)	Calling this method using the "with" prefix will ensure that all enclosed operations are sent to the CueStation window as a single operation. This is more efficient, and avoids potential transient display artifacts caused by separate GUI updates. The `SetupGUI()` and `GUIControlChanged()` callbacks are automatically put into a batch anyway, but you may want to use this function manually when manipulating the GUI from other contexts.
SendMessageToGUI(self, msg)	Sends an arbitrary Message to the host CueStation window. It's not usually necessary to call this method directly, as the other GUIClient methods will call it for you.
RemoveWidgets(self, name)	Remove one or more widgets from the custom GUI. The widget(s) to remove are specified by name; wildcards are supported. Note that any widgets contained inside the removed widgets will also be removed.
SetWindowMode(self, flags)	Sets the window-mode of the host CueStation window (to WINDOW_MODE_FLAG_MINIMIZED, WINDOW_MODE_FLAG_FULLSCREEN, WINDOW_MODE_FLAG_HIDDEN, etc). See templates/_guiscripts/test_window.py for an example of this.
SetWindowTitle(self, title)	Sets the window's title to the specified string.
SetWindowGeometry(self, left, top, width, height)	Sets the on-screen position of the host CueStation window to the specified left/top/width/height. Any parameters passed as None will be left unchanged.
InvokeMenuItem(self, menuPath, onlyIfCheckedDelta=0)	Invokes a menu item in the host CueStation window's GUI, as if the user had selected it. For example, `self.InvokeMenuItem` ("Windows/Inputs") would cause an Inputs window to be opened. Wildcards are supported.
RestartScript(self, newScriptName=None)	Causes the host CueStation window to kill this script and restart it. Or, if a script name argument is specified, that script will be started instead.
SetListenHoldEnabled(self, **args)	Enables or disables one or more per-category listen-hold states of the CueStation window. For example: `self.SetListenHoldEnabled(Outputs=True, Inputs=False).`
SaveLayoutState(self, msg, optPath="")	Saves a Message containing layout-state information into the host Window's window-layout record. This data will passed in to future Windows' `SetupGUI()` method.

INTERACTING WITH THE SERVERS

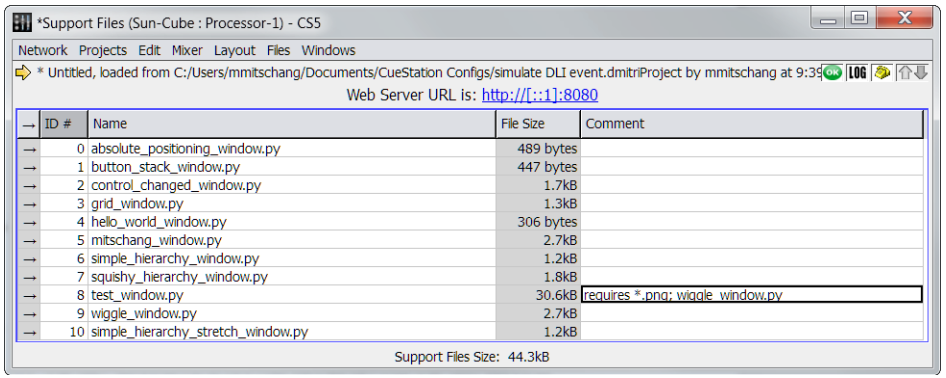
A `GUIClient` is a subclass of `BasicClient`, which means a `GUIClient` can directly subscribe to control points, upload and download information from the project database, and can perform all the same tasks that server-side Python scripts normally do. For more information about traditional D-Mitri Python functionality, see the CueStation Server-Side Python User Guide (PN 05.176.131.01). When doing traditional-style Python scripting in a client-side Python script, be aware that the `GUIClient` class does re-implement a few `BasicClient` hook methods. `ConnectedToServer()`, `SocketReadReadReady()`, and `ControlPointValueUpdated()` are implemented for its own purposes. Generally, this will not be an issue unless you decide to re-implement those methods again in your own subclass. Ensure that your re-implementation of those methods calls up to the corresponding superclass method, such as in the following script:

```
def ControlPointValueUpdated(self, address, value):
   GUIClient.ControlPointValueUpdated(self, address, value)
      [... your additional code would go here ...]
```

Otherwise, the `GUIClient` class will not receive its callback notification, and will not behave as expected.

PUTTING IT ALL TOGETHER

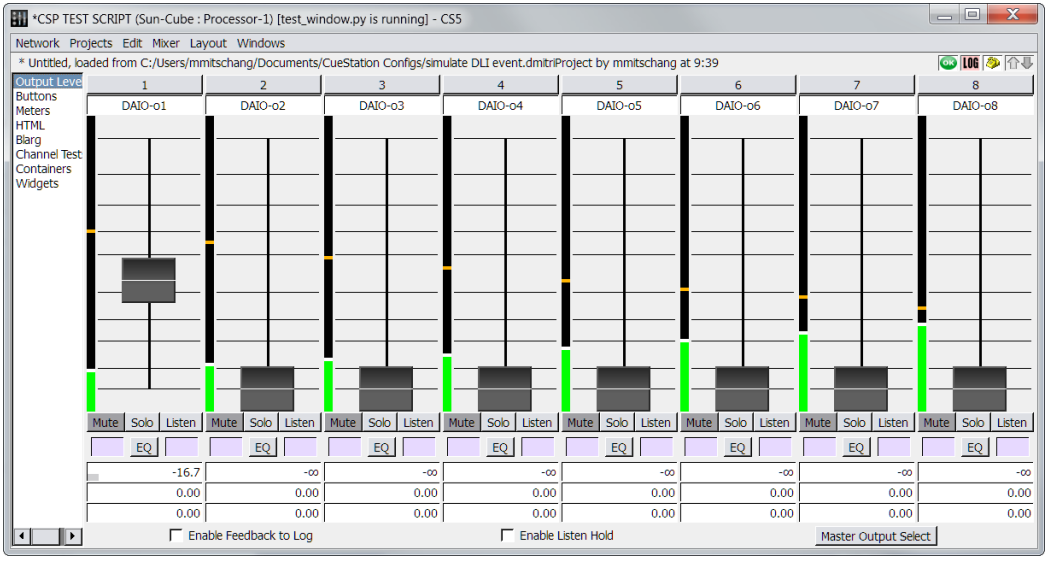
If your Python script requires other files in order to operate, you must tell the CueStation window to download those files from the Support Files area on the server before it runs your script. This is done by specifying a “requirements declaration” in the comment field of the Support Files window, as shown below:



In the above screenshot, the test_window.py script has requested that the Test Window download several additional support files before it runs the test_window.py script. Also, if any of the support files specified have their own “requires” tags, the files they require will be downloaded as well. Downloaded files will be located in a freshly created temporary folder that is unique to the host window. This folder will be automatically deleted after the script stops running. If the script needs to know the location of this temporary directory to read a file contained within, it can find out using code:

```
import os
script_dir = os.path.dirname(os.path.realpath(__file__)) + os.sep
```

Once you have an idea of how to add widgets and control them, you can assemble them together into a fully functional custom GUI, which can be as simple or as complex as desired. To see a non-trivial example of a custom GUI, run the script templates/_guiscripts/test_window.py, which displays as follows:


*CSP TEST SCRIPT (Sun-Cube : Processor-1) [test_window.py is running] - CS5
Network Projects Edit Mixer Layout Windows
* Untitled, loaded from C:/Users/mmitschang/Documents/CueStation Configs/simulate DLI event.dmitriProject by mmitschang at 9:39
Output Leve
Buttons
Meters
HTML
Blarg
Channel Test
Containers
Widgets
1 2 3 4 5 6 7 8
DAIO-o1 DAIO-o2 DAIO-o3 DAIO-o4 DAIO-o5 DAIO-o6 DAIO-o7 DAIO-o8
Mute Solo Listen
EQ
-16.7 -∞ -∞ -∞ -∞ -∞ -∞ -∞
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
Enable Feedback to Log
Enable Listen Hold
Master Output Select

APPENDIX A: INDEX TABLES

CONTAINER TYPES INDEX

Container Name	Description
frame (see “frame” on page 35)	A container widget with a customizable outline around the edges of the container.
groupbox (see “groupbox” on page 36)	A container widget with an outline around the edges of the container, plus a title string shown at the top of the container area.
scrollarea (see “scrollarea” on page 37)	An area within which a larger virtual area can be contained.
splitter (see “splitter” on page 38)	A container widget that divides its area up linearly amongst its children. The user can drag the dividers to adjust the space allocation.
tabbedarea (see “tabbedarea” on page 39)	A container widget that includes a tab bar at the top. The user can click tabs to choose which child widget to view.
widget (see “widget” on page 40)	A generic, empty widget. Can be used as a container, or as empty space.
widgetframe (see “widgetframe” on page 44)	Similar to a frame, but also supports the logic for cut-and-paste and drag-and-drop of its child widgets as a group.
widgetgroup (see “widgetgroup” on page 44)	Similar to a widget, but also supports the logic for cut-and-paste and drag-and-drop of its child widgets as a group.
widgetgroupbox (see “widgetgroupbox” on page 45)	Similar to a groupbox, but also supports the logic for cut-and-paste and drag-and-drop of its child widgets as a group.
widgetstack (see “widgetstack” on page 46)	A container widget that arranges its children into a conceptual “stack”. Only one child widget will be visible at a time.

CONTROL TYPES INDEX

Control Name	Description
busassignsview (see “busassignsview” on page 47)	Two side-by-side columns of blue bus-assign buttons.
channeltestview (see “channeltestview” on page 48)	A view with various controls for conducting channel tests.

Control Name	Description
chatentrytable (see “chatentrytable” on page 49)	A table that shows recent chat text by CueStation 5 users, as seen in the Chat window.
checkbox (see “checkbox” on page 51)	A checkbox widget with an optional text label, useful for interactive control of the Python script.
chooser (see “chooser” on page 53)	A combo box that lets the user make selections from a set of objects in a part of the project database.
clicklabel (see “clicklabel” on page 55)	A label that also responds to clicks or double-clicks.
colorpane (see “colorpane” on page 57)	A view that allows the selection of an RGB color from a palette, or via sliders.
combobox (see “combobox” on page 58)	A combo-box widget that allows the user to choose from a list of options via a pop-up menu.
cp_assignablebutton (see “cp_assignablebutton” on page 60)	This button can be used to monitor and/or control an individual bus assign control point.
cp_channellabeltext (see “cp_channellabeltext” on page 62)	A cp_text that has been customized to monitor and/or set the value of a channel label.
cp_checkbox (see “cp_checkbox” on page 64)	A checkbox that subscribes to a control point.
cp_chooser (see “cp_chooser” on page 66)	Lets the user set a control point by selecting from amongst a set of items in part of the project database.
cp_colorpane (see “cp_colorpane” on page 68)	Lets the user monitor and/or set a color control point, using a palette and/or RGB sliders.
cp_combobox (see “cp_combobox” on page 69)	Lets the user select from a number of different possible values for a specified control point.
cp_decibelfader (see “cp_decibelfader” on page 71)	A fader widget that is customized to monitor and/or set a control point whose value is expressed in decibels.
cp_decibelsmeter (see “cp_decibelsmeter” on page 72)	A meter widget that is customized to monitor a control point whose value is expressed in decibels.
cp_eqbutton (see “cp_eqbutton” on page 73)	A button whose background color specifies the state of an EQ channel, and when pressed, opens the appropriate Processing window.
cp_extralabeltext (see “cp_extralabeltext” on page 75)	The same as a cp_channellabeltext widget, except that it will exclude itself from drag-and-drop operations.

Control Name	Description
cp_fader (see “cp_fader” on page 77)	A generic fader widget.
cp_knob (see “cp_knob” on page 78)	A circular knob that can be used for monitoring and/or setting a control point.
cp_label (see “cp_label” on page 79)	A read-only text widget for monitoring the value of a control point.
cp_labelleddecibelsmeter (see “cp_labelleddecibelsmeter” on page 81)	A meter widget that displays a decibel meter value, compression/expansion value, and a superimposed text label showing the channel name.
cp_labelledmeter (see “cp_labelledmeter” on page 82)	A meter widget that shows a specified control point value as a one-item bar graph with a text label. It is used as the basis for most of the other meter widgets.
cp_lineedit (see “cp_lineedit” on page 84)	Allows the user to monitor and/or edit a string control point value as a one-line text string.
cp_listentogglebutton (see “cp_listentogglebutton” on page 86)	A toggle-button widget that knows how to handle the unification logic behind Listen control points.
cp_masterselecttogglebutton (see “cp_masterselecttogglebutton” on page 88)	A button that lights up when any of its associated slave control points is set to true. When clicked, this button sets all of its associated slave control points to the same value it has been set to.
cp_meter (see “cp_meter” on page 90)	A simple read-only widget for displaying a control point as a one-item bar graph. Note that this implementation has no text label.
cp_pagelabeltogglebutton (see “cp_pagelabeltogglebutton” on page 91)	Displays the current column number as its label. It is commonly used for channel selection.
cp_selectwatchertext (see “cp_selectwatchertext” on page 93)	A specialization of cp_text that knows to darken its background slightly when it is associated with a selected row or column.
cp_setbutton (see “cp_setbutton” on page 95)	A button that, when clicked, sets a specified control point to a specified value.
cp_solotogglebutton (see “cp_solotogglebutton” on page 97)	A specialization of cp_togglebutton that knows how to turn its background color green (or blinking-pink) depending on the state of other solo control points.
cp_text (see “cp_text” on page 99)	A widget that allows the user to monitor or quickly edit a text string control point.

Control Name	Description
cp_togglebutton (see “cp_togglebutton” on page 100)	A widget that allows the user monitor a Boolean control point, or toggle it on and off, by clicking on a button that will remain depressed until it is clicked again.
cp_vgroupbardecibelfader (see “cp_vgroupbardecibelfader” on page 102)	A specialization of the cp_decibelfader widget that also shows colored vgroup-level bars behind the fader, when appropriate.
cp_vgroupindextext (see “cp_vgroupindextext” on page 103)	A customized version of the cp_vgroupsmonitortext widget that goes blank when its value is negative.
cp_vgroupsmonitortext (see “cp_vgroupsmonitortext” on page 105)	A specialization of the cp_text widget that also knows how to color its background to give an indication of the current state of the associated VGroups, if any.
cueentrytable (see “cueentrytable” on page 107)	Shows a list of cue entries that are currently in a Cue List.
cuelibrarytable (see “cuelibrarytable” on page 109)	Shows the list of cues that are in the current project.
dial (see “dial” on page 111)	A dial widget. It works like a slider, except round.
doublespinbox (see “doublespinbox” on page 113)	A specialized spinbox that can handle floating-point values.
eqgraph (see “eqgraph” on page 115)	An EQ display graph, as seen in the Processing windows.
filedialog (see “filedialog” on page 116)	A file dialog to allow the user to select a file or directory from the local filesystem.
label (see “label” on page 118)	Shows a specified text string or image.
lcdnumber (see “lcdnumber” on page 120)	Displays a specified number using LCD-like digits.
lineedit (see “lineedit” on page 121)	Allows the user to edit a specified text string.
linkstatediagram (see “linkstatediagram” on page 123)	Shows the state of the system as a graphical diagram.
list (see “list” on page 124)	Shows a list of text strings, and allows the user to choose one of them.

Control Name	Description
logentrytable (see “logentrytable” on page 126)	Shows the current state of the system log.
menudialog (see “menudialog” on page 128)	A pop-up menu dialog that allows the user to select an item from a list. Single use only (it deletes itself when the user selects an item).
messagedialog (see “messagedialog” on page 129)	A message dialog that presents a text message to the user, and allows the user to choose a response by clicking a button. Single use only (it deletes itself when the user clicks a button).
pagearea (see “pagearea” on page 130)	Similar to a widget, but can also function as a page controller to auto-update page settings on other widgets.
pagelabel (see “pagelabel” on page 131)	A label widget that updates its label to reflect its current row/column number.
plaintextedit (see “plaintextedit” on page 133)	A multi-line text-editing widget optimized for plain text.
progressbar (see “progressbar” on page 135)	A progress bar widget, for showing the percent-completed state of a task.
pushbutton (see “pushbutton” on page 136)	A momentary-pushable button, for initiating an action.
radiobutton (see “radiobutton” on page 138)	A rounded button, typically used to select an item from a set of choices.
scrollbar (see “scrollbar” on page 140)	A scroll bar widget.
slider (see “slider” on page 141)	A widget that the user can slide to choose any value between a minimum and maximum value.
spinbox (see “spinbox” on page 143)	A widget that the user can use to edit, increment, or decrement a value between a minimum and maximum value.
statusicon (see “statusicon” on page 145)	A large widget that indicates the current state of the system (as seen in the upper right corner of the System Status window).
statuslist (see “statuslist” on page 147)	A list of modules and their current status data (as seen in the System Status window).
stringdialog (see “stringdialog” on page 149)	An input dialog that presents a text message to the user, and allows the user to enter a string value in response. Single use only (it deletes itself when the user dismisses the dialog).

Control Name	Description
subcueeditor (see “subcueeditor” on page 150)	A widget that can be used to edit a specified subcue.
subcueentrytable (see “subcueentrytable” on page 151)	A table that shows the list of subcue entries within a cue.
subcuelibrarytable (see “subcuelibrarytable” on page 154)	A table that shows the list of subcues in the project.
tabbar (see “tabbar” on page 156)	A widget that contains a series of tabs, of which the user can select one.
textbrowser (see “textbrowser” on page 157)	A multi-line rich-text browser, that supports hypertext navigation.
textedit (see “textedit” on page 160)	A multi-line text-editor/text-viewer widget, that supports display and editing of plain text and rich text.
timeleftbutton (see “timeleftbutton” on page 162)	A momentary button whose label updates itself to the number of seconds left in the current cue recall, if any. (As seen in the GO button in the Transport window)
toolbutton (see “toolbutton” on page 164)	A minimal button that takes up a bit less space than a pushbutton.
transportview (see “transportview” on page 166)	Shows a cue list with yellow and green bars, as seen in the Transport Window.
vrasdelaysgraph (see “vrasdelaysgraph” on page 167)	A graph of the VRAS Delays for a given VRAS unit.
vraserdampinggraph (see “vraserdampinggraph” on page 168)	A graph of the VRAS Early Reflection Damping for a given unit.
vrasgraph (see “vrasgraph” on page 169)	A graph of the VRAS transform for a given VRAS unit.

APPENDIX B: CONTAINER TYPES

frame

A container widget with a customizable outline around the edges of the container.

Properties

Property Name	Data Type	Usage	Description
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget's frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame's rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
lineWidth	integer	Add, Set	The width of the frame's rectangle.
midLineWidth	integer	Add, Set	The width of the frame's mid-line.

Methods

This widget type has only the standard widget methods (see "widget" on page 40).

groupbox

A container widget with an outline around the edges of the container, plus a title string shown at the top of the container area.

Properties

Property Name	Data Type	Usage	Description
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
flat	True or False	Add, Set	If True, the button will be flat in appearance; if False, the button will have beveled edges to give it a semi-3D look. Defaults to False.
title	string	Add, Set	Specifies the text to be displayed in the title of the widget.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

scrollarea

An area within which a larger virtual area can be contained. Scroll bars are supplied to allow the user to scroll around in the virtual area.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
widgetResizable	True or False	Add, Set	True if the widget the scroll area is scrolling should be resizable; False if it should be treated as a fixed-size widget.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

splitter

A container widget that divides its area up linearly amongst its children. The user can drag the dividers to adjust the space allocation.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
childrenCollaps-ible	True or False	Add, Set	True if it should be possible to collapse child nodes in a tree widget; False if the user should not be allowed to collapse them.
divpos	integer	Add, Set	An offset (in pixels) indicating how far from the top/left of the container area a splitter widget should be placed at.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
handleWidth	integer	Add, Set	The width of the splitter’s handle, in pixels.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
opaqueResize	True or False	Add, Set	True if the splitter should redraw child widgets during resize operations; False if it should not.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

tabbedarea

A container widget that includes a tab bar at the top. The user can click tabs to choose which child widget to view.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
currentIndex	integer	Add, Set	The current index that the widget is set to. Valid values range from 0 to (number of Items -1).
documentMode	True or False	Add, Set	True if the tabs should be rendered in document style; False (the default) if they should be rendered in the normal style.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for icons in the tab widgets.
movable	True or False	Add, Set	True if the user should be allowed to move the tabs within the tab bar; False (the default) otherwise.
tabPosition	TabPosition	Add, Set	Specifies the position of the tabs relative to the child widgets. Defaults to “North.”
tabShape	TabShape	Add, Set	Specifies the shape of the tabs. Defaults to `“Rounded”`.
tabsClosable	True or False	Add, Set	True if the tabs should contain close-tab buttons, False if they should not. Defaults to False.
usesScrollButtons	True or False	Add, Set	True if the tabs-strip should allow scrolling if there are too many tabs to display at once. Defaults to True.

Methods

This widget type has only the standard widget methods (see “widget” on page 40)..

widget

A generic, empty widget. Can be used as a container, or as empty space. This is the widget type you will get by default if you do not specify a type tag in your AddWidget() call.

Properties

Property Name	Data Type	Usage	Description
acceptDrops	True or False	Add, Set	True if the widget should accept drag-and-drop drop events; False if it should not.
alignment	string	Add, Set	How this widget should align its contents.
autoFillBack-ground	True or False	Add, Set	True if the widget should fill its background by default, False to leave the background fill to the parent widget.
baseSize	(width,height)	Add, Set	Used to calculate a proper widget size if the widget defines the sizeIncrement property.
bgColor	string	Add, Set	Background color for the widget. May be an English color name like “blue” or an RGB triplet string, in parentheses. For example, `(128,200,255)`.
boxAlignment	string	Add	An alignment parameter to pass to a vertical or horizontal layout-manager when adding this widget.
category	string (category)	Add	A string indicating a category for the widget to be associated with. For example `"input"`, `"output"`, `"bus"`, or `"aux"`.
col	integer	Add	Integer specifying which column this widget should be added to within a grid-layout. For example, `col=0` would indicate that this widget should go in the left-most column. It’s also possible to specify a range of columns that this widget should span, such as `col=(0-4)`.
colSpacing	integer	Add	An integer specifying how many pixels of empty space should be left between adjacent columns in a grid layout.
enabled	True or False	Add, Set	True if the widget should be enabled; False if it should be disabled. Defaults to True.

Property Name	Data Type	Usage	Description
fgColor	string	Add, Set	Foreground color for the widget. May be an English color name (like “blue”) or an RGB triplet string, in parentheses. For example, `(128,200,255)`.
fixedHeight	integer	Add, Set	If the widget should have a fixed/constant height, this property can be used to set that height.
fixedSize	(width,height)	Add, Set	If the widget should have a fixed/constant height and width, this property can be used to set those. For example, `fixedSize=(500,200)`. This is equivalent to specifying both `fixedHeight` and `fixedWidth` properties.
fixedWidth	integer	Add, Set	If the widget should have a fixed/constant width, this property can be used to set that width.
font	string (fontname)	Add, Set	The name of a font to use for this widget and its children, rather than the default font. For example, `font="courier"`.
geometry	(left,top,width, height)	Add, Set	A (left,top,width,height) set that indicates where to place this child widget, such as `geometry=(50,50,200,100)`. Useful if you want to use hard-coded widget positioning rather than layout-manager logic.
gridAlignment	string	Add	An alignment parameter to pass to a grid layout-manager when adding this widget.
horizontal	True or False	Add	True if the meter/fader should have a horizontal direction of travel; False if it should be vertical.
index	integer	Add	Description of property [index] in class [widget] needs to go here
layout	string	Add	The layout manager to use for this container widget.
maximumHeight	integer	Add, Set	The maximum height this widget may be resized to. The widget will never be resized taller than this many pixels.
maximumSize	(width,height)	Add, Set	The maximum (width,height) that this widget may be resized to. Equivalent to specifying both `maximumWidth` and `maximumHeight`.
maximumWidth	integer	Add, Set	The maximum width this widget may be resized to. The widget will never be resized wider than this many pixels.

Property Name	Data Type	Usage	Description
minimumHeight	integer	Add, Set	The minimum height this widget may be resized to. The widget will never be resized taller than this many pixels.
minimumSize	(width,height)	Add, Set	The minimum (width,height) that this widget may be resized to. Equivalent to specifying both `minimumWidth` and `minimumHeight`.
minimumWidth	integer	Add, Set	The minimum width this widget may be resized to. The widget will never be resized wider than this many pixels.
objectName	string	Add, Set	Sets the Qt internal name for this object. Note that this is NOT the same thing as the name that the Python uses to specify widgets that it previously added!
pager	string	Add, Set	Name of the widget to use as a page-controller for this widget. Not currently implemented.
parent	string	Add	Name of the widget that this widget should be added to. Only necessary if you are not satisfied with the default parent widget for some reason.
pos	(x,y)	Add, Set	Position of this widget with respect to its parent. For example, `pos=(50,100)`.
push	True	Add, Set	If `push=True` is specified as part of an `AddWidget()` call, the widget will be automatically pushed onto the parent-widget stack. However, it's better in most cases to use the `self.AddContainer(...):` idiom instead, to ensure the widget gets popped off the stack again at the right time.
row	integer	Add	Integer specifying which row this widget should be added to within a grid-layout. For example, `row=0` would indicate that this widget should go in the top-most row. It's also possible to specify a range of rows that this widget should span. For example, `row=(0-4)`.
rowSpacing	integer	Add	An integer specifying how many pixels of empty space should be left between adjacent rows in a grid layout.
size	(width,height)	Add, Set	(Width,height) indicating the size of the widget in pixels.
sizeConstraint	string (sizeConstraint)	Add	A sizing constraint passed to the parent widget's layout manager.

Property Name	Data Type	Usage	Description
sizeIncrement	(width,height)	Add, Set	Specifies the steps in which the widget's size may be increased. See also the `baseSize` property.
sizePolicy	string (sizepolicies)	Add, Set	A size policy string to help govern how the widget should be resized. Specify two values to have different policies for the horizontal and vertical directions, respectively. For example, `sizePolicy="preferred,ignored"`.
spacing	integer	Add	Specifies how many pixels should be included in a widget of type "spacing". For example, `self.AddWidget(type="spacing", spacing=15)`
stretch	integer	Add	Specifies the stretch-level of a widget of type "stretch". For Example, `self.AddWidget(type="stretch", stretch=1)`
toolTip	string	Add, Set	Specifies the text to be displayed in the tool-tip of the widget (when the mouse is hovered over the widget).
type	string	Add	Specifies the type of widget that `AddWidget()` or `AddContainer()` should create.
updatesEnabled	True or False	Add, Set	True if the widget should repaint itself when necessary; False to disable repainting. Defaults to True. Disabling updates on a widget also disables them on all its children.
vertical	True or False	Add	True if the widget should be vertical; False if the widget should be horizontal.
visible	True or False	Add, Set	True if the widget should be visible; False if it should be hidden.
whatsThis	string	Add, Set	Sets the "What's This?" text for the widget.

Methods

Method Name	Description
hide()	Hides the widget (and any child widgets it may have).

Method Name	Description
lower()	Lowers the widget's display-ordering (so that any sibling widgets that overlap this widget will partially occlude it).
raise()	Raises the widget's display-ordering (so that any sibling widgets that overlap this widget will be partially occluded by it)
show()	Shows the widget if it is not already visible, the opposite of `hide()`.
update()	Forces a repaint of the widget.

widgetframe

Similar to a frame, but also supports the logic for cut-and-paste and drag-and-drop of its child widgets as a group.

Properties

All of the standard widget properties (see "widget" on page 40), plus:

Property Name	Data Type	Usage	Description
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget's frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame's rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
lineWidth	integer	Add, Set	The width of the frame's rectangle.
midLineWidth	integer	Add, Set	The width of the frame's mid-line.

Methods

This widget type has only the standard widget methods (see "widget" on page 40).

widgetgroup

Similar to a widget, but also supports the logic for cut-and-paste and drag-and-drop of its child widgets as a group.

Properties

This widget type has only the standard widget properties (see “widget” on page 40).

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

widgetgroupbox

Similar to a groupbox, but also supports the logic for cut-and-paste and drag-and-drop of its child widgets as a group.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
flat	True or False	Add, Set	If True, the button will be flat in appearance; if False, the button will have beveled edges to give it a semi-3D look. Defaults to False.
title	string	Add, Set	Specifies the text to be displayed in the title of the widget.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

widgetstack

A container widget that arranges its children into a conceptual ”stack”. Only one child widget will be visible at a time. This container is useful when you want the GUI to switch efficiently between different modes, without having to hide and show a lot of widgets individually.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
currentIndex	integer	Add, Set	The current index that the widget is set to. Valid values range from 0 to (numberOfItems-1).
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

APPENDIX C: CONTROL TYPES

busassignsview

Two side-by-side columns of blue bus-assign buttons. This view is commonly seen in the Inputs window.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

channeltestview

A view with various controls for conducting channel tests. Channel Test Views are commonly seen at the top of Masters and Meters windows, although they are not displayed by default.

Properties

This widget type has only the standard widget properties (see “widget” on page 40).

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

chatentrytable

A table that shows recent chat text by CueStation 5 users, as seen in the Chat window.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
alternatingRowColors	True or False	Add, Set	True for alternating colors, False otherwise
autoScroll	True or False	Add, Set	True if the table should auto-scroll to assist in drag-and-drop operations. Only works if drops are enabled for the table.
autoScrollMargin	integer	Add, Set	How close (in pixels) the mouse pointer needs to be to the margin of the table for auto-scroll to be initiated.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
horizontalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll horizontally.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
selectionBehavior	SelectionBehavior	Add, Set	Governs how items may be selected.
selectionMode	SelectionMode	Add, Set	Governs how (and if) multiple items may be selected.
showGrid	True or False	Add, Set	True if the table’s between-cells grid should be visible; False if it should not be. Defaults to True.
verticalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll vertically.
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clearSelection()	De-selects any items that the widget may currently have selected.
hideColumn(int)	Hides the specified column
hideRow(int)	Hides the specified row
reset()	Resets the view to its default state. (Use with caution!)
resizeColumnToContents(int)	Resizes the nth column to fit the column’s current contents.
resizeColumnsToContents()	Resizes all columns to fit their current contents.
resizeRowToContents(int)	Resizes the nth row to fit the row’s current contents.
resizeRowsToContents()	Resizes all rows to fit their current contents.
scrollToBottom()	Scrolls to the bottom of the view.
scrollToTop()	Scrolls to the top of the view.
selectAll()	Select all of the widget’s contents.
showColumn(int)	Shows the specified column, if it was previously hidden.
showRow(int)	Shows the specified row, if it was previously hidden.
sortByColumn(int)	Sorts the table according to the data in the specified column

checkbox

A checkbox widget with an optional text label, useful for interactive control of the Python script. For a checkbox control that tracks and controls a control point, see “cp_checkbox” on page 64.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if autoRepeat is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (file-name)	Add, Set	File name of an image file (in the Support Files win-dow) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
text	string	Add, Set	Specifies the text to be displayed by the widget.
tristate	True or False	Add, Set	True if the checkbox should have three states (Checked, Unchecked, and Do Not change); False if it should have just two states (Checked and Unchecked). Defaults to False.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off

chooser

A combo box that lets the user make selections from a set of objects in a part of the project database.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoCompletion	True or False	Add, Set	True to enable autocompletion, False otherwise.
currentIndex	integer	Add, Set	The current index that the widget is set to. Valid values range from 0 to (numberOfItems-1).
editable	True or False	Add, Set	True if the widget should be editable; False if it should be read-only.
frame	True or False	Add, Set	True if the widget should have a frame-rectangle drawn around it; False otherwise.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
items	string	Add, Set	A list of strings to populate the combo box with, separated by bars. For example: `Item 1 | Item 2 | Item 3`.
minimumContentsLength	integer	Add, Set	The minimum number of characters that should fit inside the combo box. Used to set the combo box’s minimum size hint.
sizeAdjustPolicy	SizeAdjustPolicy	Add, Set	Determines how the size of the combo box changes when the content changes.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
addItem(QString)	Add a string to the widget.
clear()	Clears the widget’s content.
insertItem(int,QString)	Inserts the specified HTML into the document at the specified offset.

Method Name	Description
insertSeparator(int)	Inserts a separator dash into the widget's item list.
removeItem(int)	Removes the nth item from the items list.

clicklabel

A label that also responds to clicks or double-clicks. As seen in the left-hand column of the Input Window and the various Masters windows.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
indent	integer	Add, Set	The label’s text-indent width, in pixels. Defaults to -1, meaning that an appropriate indent will be calculated algorithmically.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
margin	integer	Add, Set	The distance (in pixels) between the innermost pixel of the frame and the outermost pixel of the contents. Defaults to 0.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
openExternalLinks	True or False	Add, Set	True if the HTML editor should open the default web browser to show links the user clicked on; False if the linked documents should appear in the widget itself.
pixmap	string (filename)	Add, Set	Name of an image file in the Support Files window that should be loaded and displayed in this widget.
scaledContents	True or False	Add, Set	True if the label’s contents should be rescaled as the label is resized; False if they should be cropped instead.
text	string	Add, Set	Specifies the text to be displayed by the widget.
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clear()	Clears the widget’s content.

colorpane

A view that allows the selection of an RGB color from a palette, or via sliders.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
color	string (colorname)	Add, Set	Which color the `colorpane` is currently specifying.
flat	True or False	Add, Set	If True, the button will be flat in appearance; if False, the button will have beveled edges to give it a semi-3D look. Defaults to False.
text	string	Add, Set	Specifies the text to be displayed by the widget.
title	string	Add, Set	Specifies the text to be displayed in the title of the widget.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

combobox

A combo-box widget that allows the user to choose from a list of options via a pop-up menu. Note that you will need to populate the combo box via one or more calls to self.InvokeWidgetMethod("my_combo_name", "addItem", "itemText") in order for it to be useful. For a combobox control that tracks and controls a control point, see cp_combobox instead.

Properties

All of the standard widget properties (see "widget" on page 40), plus:

Property Name	Data Type	Usage	Description
autoCompletion	True or False	Add, Set	True to enable autocompletion, False otherwise.
currentIndex	integer	Add, Set	The current index that the widget is set to. Valid values range from 0 to (numberOfItems-1).
editable	True or False	Add, Set	True if the widget should be editable; False if it should be read-only.
frame	True or False	Add, Set	True if the widget should have a frame-rectangle drawn around it; False otherwise.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
items	string	Add, Set	A list of strings to populate the combo box with, separated by bars. For example: `Item 1 | Item 2 | Item 3`.
minimumContentsLength	integer	Add, Set	The minimum number of characters that should fit inside the combo box. Used to set the combo box's minimum size hint.
sizeAdjustPolicy	SizeAdjustPolicy	Add, Set	Determines how the size of the combo box changes when the content changes.

Methods

All of the standard widget methods (see "widget" on page 40), plus:

Method Name	Description
addItem(QString)	Add a string to the widget.
clear()	Clears the widget's content.

Method Name	Description
insertItem(int,QString)	Inserts the specified HTML into the document at the specified offset.
insertSeparator(int)	Inserts a separator dash into the widget's item list.
removeItem(int)	Removes the nth item from the items list.

cp_assignablebutton

This button can be used to monitor and/or control an individual bus assign control point. To handle an entire channel's worth of bus assign controls at once, see buassignsview instead.

Properties

All of the standard widget properties (see "widget" on page 40), plus:

Property Name	Data Type	Usage	Description
autoRaise	True or False	Add, Set	If True, the button will be appear to be raised (via beveling) only when the mouse is hovered over it. Defaults to False.
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if `autoRepeat` is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase= "Input 1 Trim"`
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (filename)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
offText	string	Add, Set	The text the toggle-button should display when in its Off state.

Property Name	Data Type	Usage	Description
onText	string	Add, Set	The text the toggle-button should display when in its On state.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off

cp_channellabeltext

A cp_text that has been customized to monitor and/or set the value of a channel label. It is similar to a cp_text but it also knows to automatically set its background color appropriately to reflect its channel's fader-color.

Properties

All of the standard widget properties (see "widget" on page 40), plus:

Property Name	Data Type	Usage	Description
cachePixmaps	True or False	Add, Set	True if the text widget should cache previously rendered bitmaps for efficiency; False if it should not. Defaults to True.
colorMixRatio	float	Add, Set	A floating point value from 0.0 to 1.0, indicating how much of the text widget's parent's background color should be mixed into its own background color. Defaults to 0.0
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Name"`.
fillStyle	string	Add, Set	The style to use for the text widget's background state-indicator. Supported value flags are `"invisible"`, `"vertical"`, `"centered"`, or `"all"`.
fullRangeDragHeight	integer	Add, Set	The number of pixels worth of movement it should take to drag the text widget from its maximum value to its minimum value. Defaults to 100.
textOrientation	string	Add, Set	The direction in which the text inside the text widget should be written. Supported values are `"left-to-right"`, `"top-to-bottom"`, and `"bottom-to-top"`.
widestExpectedString	string	Add, Set	A string that serves as an example of the sort of text the widget will be displaying. The widget will try to keep itself large enough to display strings of this length.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cp_checkbox

A checkbox that subscribes to a control point. This control can be used to monitor and/or control a Boolean control point value.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if `autoRepeat` is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (filename)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Property Name	Data Type	Usage	Description
tristate	True or False	Add, Set	True if the checkbox should have three states (Checked, Unchecked, and Do Not Change); False if it should have just two states (Checked and Unchecked). Defaults to False.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off

cp_chooser

Lets the user set a control point by selecting from amongst a set of items in part of the project database. For example, the SpaceMap and Trajectory chooser comboboxes in the SpaceMap window are cp_chooser widgets.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoCompletion	True or False	Add, Set	True to enable autocompletion, False otherwise.
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
currentIndex	integer	Add, Set	The current index that the widget is set to. Valid values range from 0 to (numberOfItems-1).
editable	True or False	Add, Set	True if the widget should be editable; False if it should be read-only.
frame	True or False	Add, Set	True if the widget should have a frame-rectangle drawn around it; False otherwise.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
items	string	Add, Set	A list of strings to populate the combo box with, separated by bars. For example: `Item 1 | Item 2 | Item 3`.
minimumContentsLength	integer	Add, Set	The minimum number of characters that should fit inside the combo box. Used to set the combo box’s minimum size hint.
sizeAdjustPolicy	SizeAdjustPolicy	Add, Set	Determines how the size of the combo box changes when the content changes.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
addItem(QString)	Add a string to the widget.
clear()	Clears the widget’s content.
insertItem(int,QString)	Inserts the specified HTML into the document at the specified offset.
insertSeparator(int)	Inserts a separator dash into the widget’s item list.
removeItem(int)	Removes the nth item from the items list.

cp_colorpane

Lets the user monitor and/or set a color control point, using a palette and/or RGB sliders.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
color	string (colorname)	Add, Set	The color currently selected by the cp_colorpane widget
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
flat	True or False	Add, Set	If True, the button will be flat in appearance; if False, the button will have beveled edges to give it a semi-3D look. Defaults to False.
text	string	Add, Set	Specifies the text to be displayed by the widget.
title	string	Add, Set	Specifies the text to be displayed in the title of the widget.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cp_combobox

Lets the user select from a number of different possible values for a specified control point. Note that you will need to manually populate the combobox by calling via one or more calls to self.InvokeWidgetMethod("my_combo_name", "addItem", "itemText") in order for it to be useful.

Properties

All of the standard widget properties (see "widget" on page 40), plus:

Property Name	Data Type	Usage	Description
autoCompletion	True or False	Add, Set	True to enable autocompletion, False otherwise.
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
currentIndex	integer	Add, Set	The current index that the widget is set to. Valid values range from 0 to (numberOfItems-1).
editable	True or False	Add, Set	True if the widget should be editable; False if it should be read-only.
frame	True or False	Add, Set	True if the widget should have a frame-rectangle drawn around it; False otherwise.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
items	string	Add, Set	A list of strings to populate the combo box with, separated by bars. For example: `Item 1 | Item 2 | Item 3`.
minimumContentsLength	integer	Add, Set	The minimum number of characters that should fit inside the combo box. Used to set the combo box's minimum size hint.
sizeAdjustPolicy	SizeAdjustPolicy	Add, Set	Determines how the size of the combo box changes when the content changes.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
addItem(QString)	Add a string to the widget.
addState(QString,Point)	Add an item to the widget, associated with a Point value.
addState(QString,Point,QString)	Add an item to the widget, associated with a Point value, with a `toolTip` string.
addState(QString,QString)	Add An item to the widget, associated with a String value.
addState(QString,QString,QString)	Add an item to the widget, associated with a String value, with a `toolTip` string.
addState(QString,bool)	Add An item to the widget, associated with a Boolean value.
addState(QString,bool,QString)	Add an item to the widget, associated with a Boolean value, with a `toolTip` string.
addState(QString,float)	Add An item to the widget, associated with a float value.
addState(QString,float,QString)	Add an item to the widget, associated with a float value, with a `toolTip` string.
addState(QString,int32)	Add An item to the widget, associated with a int32 value.
addState(QString,int32,QString)	Add an item to the widget, associated with a int32 value, with a `toolTip` string.
addState(QString,int64)	Add An item to the widget, associated with a int64 value.
addState(QString,int64,QString)	Add an item to the widget, associated with a int64 value, with a `toolTip` string.
clear()	Clears the widget’s content.
insertItem(int,QString)	Inserts the specified HTML into the document at the specified offset.
insertSeparator(int)	Inserts a separator dash into the widget’s item list.
removeItem(int)	Removes the nth item from the items list.

cp_decibelfader

A fader widget that is customized to monitor and/or set a control point whose value is expressed in decibels.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cp_decibelsmeter

A meter widget that is customized to monitor a control point whose value is expressed in decibels.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cp_eqbutton

A button whose background color specifies the state of an EQ channel, and when pressed, opens the appropriate Processing window. This button can be seen in all Masters windows, where it is labelled ”EQ”.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoRaise	True or False	Add, Set	If True, the button will be appear to be raised (via beveling) only when the mouse is hovered over it. Defaults to False.
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if `autoRepeat` is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (filename)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off

cp_extralabeltext

The same as a cp_channellabeltext widget, except that it will exclude itself from drag-and-drop operations. This is important if you do not want a drag-and-drop operation that includes this widget to copy the channel-name control point to the target channel.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cachePixmaps	True or False	Add, Set	True if the text widget should cache previously rendered bitmaps for efficiency; False if it should not. Defaults to True.
colorMixRatio	float	Add, Set	A floating point value from 0.0 to 1.0, indicating how much of the text widget’s parent’s background color should be mixed into its own background color. Defaults to 0.0
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
fillStyle	string	Add, Set	The style to use for the text widget’s background state-indicator. Supported value flags are `"invisible"`, `"vertical"`, `"centered"`, or `"all"`.
fullRangeDragHeight	integer	Add, Set	The number of pixels worth of movement it should take to drag the text widget from its maximum value to its minimum value. Defaults to 100.
textOrientation	string	Add, Set	The direction in which the text inside the text widget should be written. Supported values are `"left-to-right"`, `"top-to-bottom"`, and `"bottom-to-top"`.
widestExpectedString	string	Add, Set	A string that serves as an example of the sort of text the widget will be displaying. The widget will try to keep itself large enough to display strings of this length.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cp_fader

A generic fader widget.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cp_knob

A circular knob that can be used for monitoring and/or setting a control point. For example it is used in the Inputs window to set Pan values.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
knobColor	string (colorname)	Add, Set	The color that the interior of the knob widget should be filled with.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cp_label

A read-only text widget for monitoring the value of a control point.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
indent	integer	Add, Set	The label’s text-indent width, in pixels. Defaults to -1, meaning that an appropriate indent will be calculated algorithmically.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
margin	integer	Add, Set	The distance (in pixels) between the innermost pixel of the frame and the outermost pixel of the contents. Defaults to 0.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
openExternalLinks	True or False	Add, Set	True if the HTML editor should open the default web browser to show links the user clicked on; False if the linked documents should appear in the widget itself.
pixmap	string (filename)	Add, Set	Name of an image file in the Support Files window that should be loaded and displayed in this widget.
scaledContents	True or False	Add, Set	True if the label’s contents should be rescaled as the label is resized; False if they should be cropped instead.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Property Name	Data Type	Usage	Description
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clear()	Clears the widget’s content.

cp_labelleddecibelsmeter

A meter widget that displays a decibel meter value, compression/expansion value, and a superimposed text label showing the channel name. These widgets can be seen in the Meters windows (at least when ”Use Hardware Graphics Acceleration” is disabled in the Layout menu. When ”Use Hardware Graphics Acceleration” is enabled, the Meters windows use an OpenGL-based metering implementation instead).

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cp_labelledmeter

A meter widget that shows a specified control point value as a one-item bar graph with a text label. It is used as the basis for most of the other meter widgets.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
indent	integer	Add, Set	The label’s text-indent width, in pixels. Defaults to -1, meaning that an appropriate indent will be calculated algorithmically.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
margin	integer	Add, Set	The distance (in pixels) between the innermost pixel of the frame and the outermost pixel of the contents. Defaults to 0.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
openExternalLinks	True or False	Add, Set	True if the HTML editor should open the default web browser to show links the user clicked on; False if the linked documents should appear in the widget itself.
pixmap	string (filename)	Add, Set	Name of an image file in the Support Files window that should be loaded and displayed in this widget.
scaledContents	True or False	Add, Set	True if the label’s contents should be rescaled as the label is resized; False if they should be cropped instead.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Property Name	Data Type	Usage	Description
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clear()	Clears the widget’s content.

cp_lineedit

Allows the user to monitor and/or edit a string control point value as a one-line text string.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
cursorPosition	integer	Add, Set	The position (in characters) of the text-editing cursor within the line-edit. Defaults to 0 (the cursor is at the beginning of the text string).
echoMode	EchoMode	Add, Set	How characters should be displayed in the line-edit widget.
frame	True or False	Add, Set	True if the widget should have a frame-rectangle drawn around it; False otherwise.
inputMask	string	Add, Set	A string that can be used to limit which characters may be entered into the lineedit widget. See Qt’s QLineEdit documentation for details.
maxLength	integer	Add, Set	Maximum length (in characters) of the string that may be entered by the user.
placeholderText	string	Add, Set	Text that should be displayed (in a grayed-out color) when the user has not yet entered any text.
readOnly	True or False	Add, Set	True if the widget should be read-only; False if it should be editable.
text	string	Add, Set	Specifies the text to be displayed by the widget.
widestExpectedString	string	Add, Set	A string that serves as an example of the sort of text the widget will be displaying. The widget will try to keep itself large enough to display strings of this length.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clear()	Clears the widget’s content.
copy()	Copies the selected contents of the widget into the global clipboard.
cut()	Cuts the selected contents of the widget into the global clipboard.
paste()	Pastes the clipboard’s current contents into the widget.
selectAll()	Select all of the widget’s contents.

cp_listentogglebutton

A toggle-button widget that knows how to handle the unification logic behind Listen control points. It can be seen just below the “U” button in Masters window columns.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
activeListenSet	string	Add, Set	String indicating which listen bus(es) should be associated with this button. For example, `1L,2,3R` would specify the left side of Listen bus 1, both sides of Listen bus 2, and the right side of Listen bus 3.
autoRaise	True or False	Add, Set	If True, the button will be appear to be raised (via beveling) only when the mouse is hovered over it. Defaults to False.
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if `autoRepeat` is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
colors	string (colors)	Add, Set	Unpressed and pressed colors for the button. May contain up to four color names, for on-state, off-state, on-state-blink, and off-state-blink, respectively. For example, `colors="grey,green,red,blue"` would cause the button to blink grey-and-red when pressed, and green-and-blue when un-pressed. Defaults will be chosen for any colors left unspecified.

Property Name	Data Type	Usage	Description
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (filename)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
offText	string	Add, Set	The text the toggle-button should display when in its Off state.
onText	string	Add, Set	The text the toggle-button should display when in its On state.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off

cp_masterselecttogglebutton

A button that lights up when any of its associated slave control points is set to true. When clicked, this button sets all of its associated slave control points to the same value it has been set to. Examples of this control can be seen at the left-hand side of the various Masters Windows.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoRaise	True or False	Add, Set	If True, the button will be appear to be raised (via beveling) only when the mouse is hovered over it. Defaults to False.
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if `autoRepeat` is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
colors	string (colors)	Add, Set	Unpressed and pressed colors for the button. May contain up to four color names, for on-state, off-state, on-state-blink, and off-state-blink, respectively. For example, `colors="grey,green,red,blue"` would cause the button to blink grey-and-red when pressed, and green-and-blue when un-pressed. Reasonable defaults will be chosen for any colors left unspecified.

Property Name	Data Type	Usage	Description
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (filename)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
offText	string	Add, Set	The text the toggle-button should display when in its Off state.
onText	string	Add, Set	The text the toggle-button should display when in its On state.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off.

cp_meter

A simple read-only widget for displaying a control point as a one-item bar graph. Note that this implementation has no text label. This widget can be seen as the meter widget in the various Masters windows.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cp_pagelabeltogglebutton

Displays the current column number as its label. It is commonly used for channel selection. It can be seen at the top of the various Masters windows, at the top of the Matrix window, and so on.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoRaise	True or False	Add, Set	If True, the button will be appear to be raised (via beveling) only when the mouse is hovered over it. Defaults to False.
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if `autoRepeat` is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
colors	string (colors)	Add, Set	Unpressed and pressed colors for the button. May contain up to four color names, for on-state, off-state, on-state-blink, and off-state-blink, respectively. For example, `colors="grey,green,red,blue"` would cause the button to blink grey-and-red when pressed, and green-and-blue when un-pressed. Reasonable defaults will be chosen for any colors left unspecified.
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.

Property Name	Data Type	Usage	Description
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (filename)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
offText	string	Add, Set	The text the toggle-button should display when in its Off state.
onText	string	Add, Set	The text the toggle-button should display when in its On state.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off

cp_selectwatchertext

A specialization of cp_text that knows to darken its background slightly when it is associated with a selected row or column. The cells in the main grid area of the Matrix window are of this type.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cachePixmaps	True or False	Add, Set	True if the text widget should cache previously rendered bitmaps for efficiency; False if it should not. Defaults to True.
colorMixRatio	float	Add, Set	A floating point value from 0.0 to 1.0, indicating how much of the text widget’s parent’s background color should be mixed into its own background color. Defaults to 0.0
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
fillStyle	string	Add, Set	The style to use for the text widget’s background state-indicator. Supported value flags are `"invisible"`, `"vertical"`, `"centered"`, or `"all"`.
fullRangeDragHeight	integer	Add, Set	The number of pixels worth of movement it should take to drag the text widget from its maximum value to its minimum value. Defaults to 100.
textOrientation	string	Add, Set	The direction in which the text inside the text widget should be written. Supported values are `"left-to-right"`, `"top-to-bottom"`, and `"bottom-to-top"`.
widestExpectedString	string	Add, Set	A string that serves as an example of the sort of text the widget will be displaying. The widget will try to keep itself large enough to display strings of this length.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cp_setbutton

A button that, when clicked, sets a specified control point to a specified value.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoRaise	True or False	Add, Set	If True, the button will be appear to be raised (via beveling) only when the mouse is hovered over it. Defaults to False.
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if `autoRepeat` is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (file-name)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off

cp_solotogglebutton

A specialization of cp_togglebutton that knows how to turn its background color green (or blinking-pink) depending on the state of other solo control points. It can be seen in the middle of the Inputs window, labelled ”S”.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoRaise	True or False	Add, Set	If True, the button will be appear to be raised (via beveling) only when the mouse is hovered over it. Defaults to False.
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if `autoRepeat` is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
colors	string (colors)	Add, Set	Unpressed and pressed colors for the button. May contain up to four color names, for on-state, off-state, on-state-blink, and off-state-blink, respectively. For example, `colors="grey,green,red,blue"` would cause the button to blink grey-and-red when pressed, and green-and-blue when un-pressed. Reasonable defaults will be chosen for any colors left unspecified.
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.

Property Name	Data Type	Usage	Description
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (filename)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
offText	string	Add, Set	The text the toggle-button should display when in its Off state.
onText	string	Add, Set	The text the toggle-button should display when in its On state.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off.

cp_text

A widget that allows the user to monitor or quickly edit a text string control point. Examples of this widget are ubiquitous in CueStation, for example at the bottom of the Input window.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cachePixmaps	True or False	Add, Set	True if the text widget should cache previously rendered bitmaps for efficiency; False if it should not. Defaults to True.
colorMixRatio	float	Add, Set	A floating point value from 0.0 to 1.0, indicating how much of the text widget’s parent’s background color should be mixed into its own background color. Defaults to 0.0
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
fillStyle	string	Add, Set	The style to use for the text widget’s background state-indicator. Supported value flags are `"invisible"`, `"vertical"`, `"centered"`, or `"all"`.
fullRangeDragHeight	integer	Add, Set	The number of pixels worth of movement it should take to drag the text widget from its maximum value to its minimum value. Defaults to 100.
textOrientation	string	Add, Set	The direction in which the text inside the text widget should be written. Supported values are `"left-to-right"`, `"top-to-bottom"`, and `"bottom-to-top"`.
widestExpectedString	string	Add, Set	A string that serves as an example of the sort of text the widget will be displaying. The widget will try to keep itself large enough to display strings of this length.

cp_togglebutton

A widget that allows the user monitor a Boolean control point, or toggle it on and off, by clicking on a button that will remain depressed until it is clicked again.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoRaise	True or False	Add, Set	If True, the button will be appear to be raised (via beveling) only when the mouse is hovered over it. Defaults to False.
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if `autoRepeat` is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
colors	string (colors)	Add, Set	Unpressed and pressed colors for the button. May contain up to four color names, for on-state, off-state, on-state-blink, and off-state-blink, respectively. For example, `colors="grey,green,red,blue"` would cause the button to blink grey-and-red when pressed, and green-and-blue when un-pressed. Reasonable defaults will be chosen for any colors left unspecified.
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.

Property Name	Data Type	Usage	Description
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (filename)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
offText	string	Add, Set	The text the toggle-button should display when in its Off state.
onText	string	Add, Set	The text the toggle-button should display when in its On state.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off

cp_vgroupbardecibelfader

A specialization of the cp_decibelfader widget that also shows colored vgroup-level bars behind the fader, when appropriate. It can be seen in any of the Masters windows (note that the colored bars only appear when a VGroup has been set for the channel).

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cp_vgroupindextext

A customized version of the cp_vgroupsmonitortext widget that goes blank when its value is negative. It is commonly used for selecting an index; for example a VGroup Index in the Masters window.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cachePixmaps	True or False	Add, Set	True if the text widget should cache previously rendered bitmaps for efficiency; False if it should not. Defaults to True.
colorMixRatio	float	Add, Set	A floating point value from 0.0 to 1.0, indicating how much of the text widget’s parent’s background color should be mixed into its own background color. Defaults to 0.0
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
fillStyle	string	Add, Set	The style to use for the text widget’s background state-indicator. Supported value flags are `"invisible"`, `"vertical"`, `"centered"`, or `"all"`.
fullRangeDragHeight	integer	Add, Set	The number of pixels worth of movement it should take to drag the text widget from its maximum value to its minimum value. Defaults to 100.
textOrientation	string	Add, Set	The direction in which the text inside the text widget should be written. Supported values are `"left-to-right"`, `"top-to-bottom"`, and `"bottom-to-top"`.
widestExpectedString	string	Add, Set	A string that serves as an example of the sort of text the widget will be displaying. The widget will try to keep itself large enough to display strings of this length.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cp_vgroupsmonitortext

A specialization of the cp_text widget that also knows how to color its background to give an indication of the current state of the associated VGroups, if any. There are two of this widget in each column in the various Masters windows, directly to the right of each fader widget.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cachePixmaps	True or False	Add, Set	True if the text widget should cache previously rendered bitmaps for efficiency; False if it should not. Defaults to True.
colorMixRatio	float	Add, Set	A floating point value from 0.0 to 1.0, indicating how much of the text widget’s parent’s background color should be mixed into its own background color. Defaults to 0.0
cpbase	string (address)	Add, Set	Address string indicating the Control Point Address this widget should associated itself with. For example, `cpbase="Input 1 Trim"`.
fillStyle	string	Add, Set	The style to use for the text widget’s background state-indicator. Supported value flags are `"invisible"`, `"vertical"`, `"centered"`, or `"all"`.
fullRangeDragHeight	integer	Add, Set	The number of pixels worth of movement it should take to drag the text widget from its maximum value to its minimum value. Defaults to 100.
textOrientation	string	Add, Set	The direction in which the text inside the text widget should be written. Supported values are `"left-to-right"`, `"top-to-bottom"`, and `"bottom-to-top"`.
widestExpectedString	string	Add, Set	A string that serves as an example of the sort of text the widget will be displaying. The widget will try to keep itself large enough to display strings of this length.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

cueentrytable

Shows a list of cue entries that are currently in a Cue List.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
alternatingRowColors	True or False	Add, Set	True for alternating colors, False otherwise
autoScroll	True or False	Add, Set	True if the table should auto-scroll to assist in drag-and-drop operations. Only works if drops are enabled for the table.
autoScrollMargin	integer	Add, Set	How close (in pixels) the mouse pointer needs to be to the margin of the table for auto-scroll to be initiated.
cueListID	integer	Add, Set	ID of the Cue List whose contents this table should display.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
horizontalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll horizontally.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
selectionBehavior	SelectionBehavior	Add, Set	Governs how items may be selected.
selectionMode	SelectionMode	Add, Set	Governs how (and if) multiple items may be selected.
showGrid	True or False	Add, Set	True if the table’s between-cells grid should be visible; False if it should not be. Defaults to True.
verticalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll vertically.

Property Name	Data Type	Usage	Description
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clearSelection()	De-selects any items that the widget may currently have selected.
hideColumn(int)	Hides the specified column
hideRow(int)	Hides the specified row
reset()	Resets the view to its default state. (Use with caution!)
resizeColumnToContents(int)	Resizes the nth column to fit the column’s current contents.
resizeColumnsToContents()	Resizes all columns to fit their current contents.
resizeRowToContents(int)	Resizes the nth row to fit the row’s current contents.
resizeRowsToContents()	Resizes all rows to fit their current contents.
scrollToBottom()	Scrolls to the bottom of the view.
scrollToTop()	Scrolls to the top of the view.
selectAll()	Select all of the widget’s contents.
showColumn(int)	Shows the specified column, if it was previously hidden.
showRow(int)	Shows the specified row, if it was previously hidden.
sortByColumn(int)	Sorts the table according to the data in the specified column

cuelibrarytable

Shows the list of cues that are in the current project.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
alternatingRowColors	True or False	Add, Set	True for alternating colors, False otherwise
autoScroll	True or False	Add, Set	True if the table should auto-scroll to assist in drag-and-drop operations. Only works if drops are enabled for the table.
autoScrollMargin	integer	Add, Set	How close (in pixels) the mouse pointer needs to be to the margin of the table for auto-scroll to be initiated.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
horizontalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll horizontally.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
selectionBehavior	SelectionBehavior	Add, Set	Governs how items may be selected.
selectionMode	SelectionMode	Add, Set	Governs how (and if) multiple items may be selected.
showGrid	True or False	Add, Set	True if the table’s between-cells grid should be visible; False if it should not be. Defaults to True.
verticalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll vertically.
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clearSelection()	De-selects any items that the widget may currently have selected.
hideColumn(int)	Hides the specified column
hideRow(int)	Hides the specified row
reset()	Resets the view to its default state. (Use with caution!)
resizeColumnToContents(int)	Resizes the nth column to fit the column’s current contents.
resizeColumnsToContents()	Resizes all columns to fit their current contents.
resizeRowToContents(int)	Resizes the nth row to fit the row’s current contents.
resizeRowsToContents()	Resizes all rows to fit their current contents.
scrollToBottom()	Scrolls to the bottom of the view.
scrollToTop()	Scrolls to the top of the view.
selectAll()	Select all of the widget’s contents.
showColumn(int)	Shows the specified column, if it was previously hidden.
showRow(int)	Shows the specified row, if it was previously hidden.
sortByColumn(int)	Sorts the table according to the data in the specified column

dial

A dial widget. It works like a slider, except round.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
invertedAppearance	True or False	Add, Set	If True, the widget’s mapping will be mirrored, such that its minimum and maximum extreme values are in opposite locations. Defaults to False.
invertedControls	True or False	Add, Set	If True, the widget’s key-mappings will be inverted (such as inverting the traditional “increase” key-strokes and mouse-wheel motions to mean “decrease,” and vice-versa).
maximum	integer	Add, Set	Maximum value allowed by the widget.
minimum	integer	Add, Set	Minimum value allowed by the widget.
notchTarget	qreal	Add, Set	The intended/ideal number of pixels that should appear between adjacent notch positions. Defaults to 3.7. Note that the actual spacing may differ from this value.
notchesVisible	True or False	Add, Set	True if notch marks should be displayed; False other-wise. Defaults to False.
pageStep	integer	Add, Set	Integer value for the size of the increase or decrease the widget’s value should undergo when the user pressed page-up or page-down.
singleStep	integer	Add, Set	Integer value for the size of the increase or decrease the widget’s value should undergo when the user presses the up or down arrow (or clicks the up or down button).
sliderDown	True or False	Add, Set	Set true if the slider should act as if it is being pressed down by the mouse button.
sliderPosition	integer	Add, Set	Current position of the slider. This is a synonym for the “value” property.

Property Name	Data Type	Usage	Description
tracking	True or False	Add, Set	If True, updates will be produced whenever the user drags the slider. If False, updates will be produced only when the user ends the drag. Defaults to True.
value	integer	Add, Set	The current value of the control.
wrapping	True or False	Add, Set	True if the dial should be allowed to wrap around from maximum to minimum values, and vice versa. Defaults to False.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

doublespinbox

A specialized spinbox that can handle floating-point values.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
accelerated	True or False	Add, Set	True if the rate-of-change of the value should increase as the user holds down the increase/decrease button over a longer time.
buttonSymbols	ButtonSymbols	Add, Set	What symbols to show in the increment/decrement buttons.
decimals	integer	Add, Set	Precision of the spin-box’s numeric display, in decimals. Valid range is 0–323.
frame	True or False	Add, Set	True if the widget should have a frame-rectangle drawn around it; False otherwise.
keyboardTracking	True or False	Add, Set	If True, value-changed updates are emitted while the user is entering a new value using the keyboard. If False, they are only emitted when the user has finished.
maximum	double	Add, Set	Maximum value allowed by the widget.
minimum	double	Add, Set	Minimum value allowed by the widget.
prefix	string	Add, Set	An optional string to display before the value string. For example, `prefix="$"` would be appropriate for displaying a dollar amount.
readOnly	True or False	Add, Set	True if the widget should be read-only; False if it should be editable.
singleStep	double	Add, Set	Integer value for the size of the increase or decrease the widget’s value should undergo when the user presses the up or down arrow (or clicks the up or down button).
specialValueText	string	Add, Set	A special value to display when the spin-box’s value is equal to its minimum value. Typically used when the minimum value has a special/default meaning.

Property Name	Data Type	Usage	Description
suffix	string	Add, Set	An optional string to display after the numeric value. For example, `suffix=" km"`.
value	double	Add, Set	The current value of the control.
wrapping	True or False	Add, Set	True if the dial should be allowed to wrap around from maximum to minimum values, and vice versa. Defaults to False.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clear()	Clears the widget’s content.
selectAll()	Select all of the widget’s contents.
stepDown()	Decrements the value by one step.
stepUp()	Increments the value by one step.

eqgraph

An EQ display graph, as seen in the Processing windows. It can graphically show the current EQ for a specified channel.

Properties

This widget type has only the standard widget properties (see “widget” on page 40).

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

filedialog

A file dialog to allow the user to select a file or directory from the local filesystem.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
acceptMode	AcceptMode	Add, Set	Description of property [acceptMode] in class [filedialog] needs to go here
defaultSuffix	string	Add, Set	Default file-type-extension suffix to give to files where the user didn’t specify a suffix.
fileMode	FileMode	Add	Mode to open the file dialog with. Supported value strings are: “`SaveFile`”, “`OpenFile`”, “`OpenFiles`”, “`OpenDirectory`”, and “`OpenDirectoryOnly`”.
modal	True or False	Add, Set	True if this dialog should prevent user interaction with other GUI elements until it is closed; False otherwise. Defaults to False.
nameFilterDetailsVisible	True or False	Add, Set	True if details about filename-filtering should be visible to the user; False if they should be hidden.
options	Options	Add, Set	Description of property [options] in class [filedialog] needs to go here
readOnly	True or False	Add, Set	True if the widget should be read-only; False if it should be editable.
resolveSymlinks	True or False	Add, Set	True if the file dialog should transparently resolve symbolic links; False if it should not.
title	string	Add, Set	Specifies the text to be displayed in the title of the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
RereadDir()	Re-read the current directory (in case it has changed)

Method Name	Description
accept()	Accept the dialog (as if the user had clicked the Okay or Accept button).
done(int)	Dismisses the dialog with the specified result-code.
open()	Tells the file dialog to open the selected file (as if the user had clicked the Open button).
reject()	Dismisses the dialog, as if the user had clicked the Cancel button.

label

Shows a specified text string or image.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
indent	integer	Add, Set	The label’s text-indent width, in pixels. Defaults to -1, meaning that an appropriate indent will be calculated algorithmically.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
margin	integer	Add, Set	The distance (in pixels) between the innermost pixel of the frame and the outermost pixel of the contents. Defaults to 0.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
openExternalLinks	True or False	Add, Set	True if the HTML editor should open the default web browser to show links the user clicked on; False if the linked documents should appear in the widget itself.
pixmap	string (filename)	Add, Set	Name of an image file in the Support Files window that should be loaded and displayed in this widget.
scaledContents	True or False	Add, Set	True if the label’s contents should be rescaled as the label is resized; False if they should be cropped instead.
text	string	Add, Set	Specifies the text to be displayed by the widget.
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clear()	Clears the widget’s content.

lcdnumber

Displays a specified number using LCD-like digits.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
digitCount	integer	Add, Set	How many digits should be displayed.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
intValue	integer	Add, Set	Integer value held by the widget.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
segmentStyle	SegmentStyle	Add, Set	How segments of the LCD digits should be rendered.
smallDecimalPoint	True or False	Add, Set	True if the decimal point should get only a little bit of space between digits; False if it should get a full digit’s worth of space to itself.
value	double	Add, Set	The current value of the control.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
display(QString)	Displays the number specified in the argument string.
display(double)	Displays the number specified in the argument.
display(int)	Displays the number specified in the argument.

lineedit

Allows the user to edit a specified text string.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
cursorPosition	integer	Add, Set	The position (in characters) of the text-editing cursor within the line-edit. Defaults to 0 (the cursor is at the beginning of the text string).
echoMode	EchoMode	Add, Set	How characters should be displayed in the line-edit widget.
frame	True or False	Add, Set	True if the widget should have a frame-rectangle drawn around it; False otherwise.
inputMask	string	Add, Set	A string that can be used to limit which characters may be entered into the lineedit widget. See Qt’s QLineEdit documentation for details.
maxLength	integer	Add, Set	Maximum length (in characters) of the string that may be entered by the user.
placeholderText	string	Add, Set	Text that should be displayed (in a grayed-out color) when the user has not yet entered any text.
readOnly	True or False	Add, Set	True if the widget should be read-only; False if it should be editable.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clear()	Clears the widget’s content.
copy()	Copies the selected contents of the widget into the global clipboard.
cut()	Cuts the selected contents of the widget into the global clipboard.

Method Name	Description
paste()	Pastes the clipboard's current contents into the widget.
selectAll()	Select all of the widget's contents.

linkstatediagram

Shows the state of the system as a graphical diagram. This widget can be seen at the top of the System Status window.

Properties

This widget type has only the standard widget properties (see “widget” on page 40).

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

list

Shows a list of text strings, and allows the user to choose one of them.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
alternatingRowColors	True or False	Add, Set	True for alternating colors, False otherwise
autoScroll	True or False	Add, Set	True if the table should auto-scroll to assist in drag-and-drop operations. Only works if drops are enabled for the table.
autoScrollMargin	integer	Add, Set	How close (in pixels) the mouse pointer needs to be to the margin of the table for auto-scroll to be initiated.
currentRow	integer	Add, Set	The current row that is selected in the list widget. Valid values range from 0 to (numberOfRows-1).
flow	Flow	Add, Set	Direction in which items should flow during layout.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
gridSize	(width,height)	Add, Set	Size of the grid in which the items are laid out. Default state is no-grid-used.
horizontalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll horizontally.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
isWrapping	True or False	Add, Set	True if the items layout should wrap when there is no more space in the visible area.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
resizeMode	ResizeMode	Add, Set	Governs how items layout will change when the view is resized.

Property Name	Data Type	Usage	Description
selectionBehavior	SelectionBehavior	Add, Set	Governs how items may be selected.
selectionMode	SelectionMode	Add, Set	Governs how (and if) multiple items may be selected.
selectionRectVisible	True or False	Add, Set	True if the selection-rectangle should be made visible when the user is dragging out a rectangle to select items. Has no effect when the widget is in single-selection mode. Defaults to False.
verticalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll vertically.
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
addItem(QString)	Add a string to the widget.
clear()	Clears the widget’s content.
clearSelection()	De-selects any items that the widget may currently have selected.
insertItem(int,QString)	Inserts the specified HTML into the document at the specified offset.
removeItem(int)	Removes the nth item from the items list.
reset()	Resets the view to its default state. (Use with caution!)
scrollToBottom()	Scrolls to the bottom of the view.
scrollToTop()	Scrolls to the top of the view.
selectAll()	Select all of the widget’s contents.

logentrytable

Shows the current state of the system log.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
alternatingRowColors	True or False	Add, Set	True for alternating colors, False otherwise
autoScroll	True or False	Add, Set	True if the table should auto-scroll to assist in drag-and-drop operations. Only works if drops are enabled for the table.
autoScrollMargin	integer	Add, Set	How close (in pixels) the mouse pointer needs to be to the margin of the table for auto-scroll to be initiated.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
horizontalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll horizontally.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
selectionBehavior	SelectionBehavior	Add, Set	Governs how items may be selected.
selectionMode	SelectionMode	Add, Set	Governs how (and if) multiple items may be selected.
showGrid	True or False	Add, Set	True if the table’s between-cells grid should be visible; False if it should not be. Defaults to True.
verticalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll vertically.
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clearSelection()	De-selects any items that the widget may currently have selected.
hideColumn(int)	Hides the specified column
hideRow(int)	Hides the specified row
reset()	Resets the view to its default state. (Use with caution!)
resizeColumnToContents(int)	Resizes the nth column to fit the column’s current contents.
resizeColumnsToContents()	Resizes all columns to fit their current contents.
resizeRowToContents(int)	Resizes the nth row to fit the row’s current contents.
resizeRowsToContents()	Resizes all rows to fit their current contents.
scrollToBottom()	Scrolls to the bottom of the view.
scrollToTop()	Scrolls to the top of the view.
selectAll()	Select all of the widget’s contents.
showColumn(int)	Shows the specified column, if it was previously hidden.
showRow(int)	Shows the specified row, if it was previously hidden.
sortByColumn(int)	Sorts the table according to the data in the specified column

menudialog

A pop-up menu dialog that allows the user to select an item from a list. Single use only (it deletes itself when the user selects an item).

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
icon	string (filename)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
separatorsCollapsible	True or False	Add, Set	Governs whether consecutive separators should be collapsed into a single separator. Defaults to True.
title	string	Add, Set	Specifies the text to be displayed in the title of the widget.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

messagedialog

A message dialog that presents a text message to the user, and allows the user to choose a response by clicking a button. Single use only (it deletes itself when the user clicks a button).

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
detailedText	string	Add, Set	The text to display in the details area.
icon	Icon	Add, Set	A string indicating which built-in icon to display in the dialog.
iconPixmap	string (file-name)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
informativeText	string	Add, Set	Some additional text that can be displayed as a supplement to the messagedialog’s normal text.
modal	True or False	Add, Set	True if this dialog should prevent user interaction with other GUI elements until it is closed; False otherwise. Defaults to False.
standardButtons	StandardButtons	Add, Set	Specifies which button(s) should be available in the dialog for the user to click.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
accept()	Accept the dialog (as if the user had clicked the Okay or Accept button).
done(int)	Dismisses the dialog with the specified result-code.
open()	Tells the file dialog to open the selected file (as if the user had clicked the Open button).
reject()	Dismisses the dialog, as if the user had clicked the ”Cancel” button.

pagearea

Similar to a widget, but can also function as a page controller to auto-update page settings on other widgets.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
enablePaging	True or False	Add, Set	True if paging should be enabled; False if all widgets being paged by this pagearea should be temporarily disabled/unmapped. Defaults to True.
page0	integer	Add, Set	Offset for tracking paged widgets to use for their first paging axis. Defaults to zero.
page1	integer	Add, Set	Offset for tracking paged widgets to use for their second paging axis. Defaults to zero.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

pagelabel

A label widget that updates its label to reflect its current row/column number.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
indent	integer	Add, Set	The label’s text-indent width, in pixels. Defaults to -1, meaning that an appropriate indent will be calculated algorithmically.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
margin	integer	Add, Set	The distance (in pixels) between the innermost pixel of the frame and the outermost pixel of the contents. Defaults to 0.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
openExternalLinks	True or False	Add, Set	True if the HTML editor should open the default web browser to show links the user clicked on; False if the linked documents should appear in the widget itself.
pixmap	string (file-name)	Add, Set	Name of an image file in the Support Files window that should be loaded and displayed in this widget.
scaledContents	True or False	Add, Set	True if the label’s contents should be rescaled as the label is resized; False if they should be cropped instead.
text	string	Add, Set	Specifies the text to be displayed by the widget.
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clear()	Clears the widget’s content.

plaintextedit

A multi-line text-editing widget optimized for plain text.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
backgroundVisible	True or False	Add, Set	True if the palette background should be visible outside the document area. When False, the user can visually distinguish areas that are not part of the document.
centerOnScroll	True or False	Add, Set	If True, the text area will scroll vertically to keep the cursor in the vertical center of the viewport. If False, the cursor will be kept on-screen but not necessarily centered.
cursorWidth	integer	Add, Set	Width of the text-editing cursor, in pixels.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
lineWrapMode	LineWrapMode	Add, Set	Specifies how per-line text-wrapping should be handled.
maximumBlockCount	integer	Add, Set	Specifies the maximum number of blocks (paragraphs) the document may contain. Defaults to 0 (aka no limit).
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
overwriteMode	True or False	Add, Set	If True, text entered by the user will overwrite any existing text at the cursor location. If False, new text will be inserted at the cursor location.
plainText	string	Add, Set	Plain text to display in this widget.
readOnly	True or False	Add, Set	True if the widget should be read-only; False if it should be editable.

Property Name	Data Type	Usage	Description
tabChangesFocus	True or False	Add, Set	If True, the tab key will cause a focus-change to another widget. If false, the tab key will enter a tab character into the text. Defaults to False.
tabStopWidth	integer	Add, Set	Width of a tab character, in pixels.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
appendHtml(QString)	Appends the specified HTML string to the widget’s content.
appendPlainText(QString)	Appends the specified plain-text string to the widget’s content.
centerCursor()	Scrolls the viewport so that the text-editing cursor is in the center of the visible area.
clear()	Clears the widget’s content.
copy()	Copies the selected contents of the widget into the global clipboard.
cut()	Cuts the selected contents of the widget into the global clipboard.
insertPlainText(QString)	Inserts the specified plain text into the document.
paste()	Pastes the clipboard’s current contents into the widget.
selectAll()	Select all of the widget’s contents.

progressbar

A progress bar widget, for showing the percent-completed state of a task.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
format	string	Add, Set	A printf-style string for how progress-bar text should be formatted. Allowed special chars are `%p` for percent completed, `%v` for current value, or `%m` for number of steps. Defaults to “`%p%`”.
invertedAppearance	True or False	Add, Set	If True, the widget’s mapping will be mirrored, such that its minimum and maximum extreme values are in opposite locations. Defaults to False.
maximum	integer	Add, Set	Maximum value allowed by the widget.
minimum	integer	Add, Set	Minimum value allowed by the widget.
textVisible	True or False	Add, Set	True if the text in the progress bar should be shown, of False if the text should be hidden.
value	integer	Add, Set	The current value of the control.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
reset()	Resets the view to its default state. (Use with caution!)

pushbutton

A momentary-pushable button for initiating an action.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if `autoRepeat` is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
flat	True or False	Add, Set	If True, the button will be flat in appearance; if False, the button will have beveled edges to give it a semi-3D look. Defaults to False.
icon	string (file-name)	Add, Set	File name of an image file (in the Support Files win-dow) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off

radiobutton

A rounded button, typically used to select an item from a set of choices. When one radio button within a container is pressed, the other radio buttons will automatically un-press themselves.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if `autoRepeat` is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (filename)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off

scrollbar

A scroll bar widget.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
invertedAppearance	True or False	Add, Set	If True, the widget’s mapping will be mirrored, such that its minimum and maximum extreme values are in opposite locations. Defaults to False.
invertedControls	True or False	Add, Set	If True, the widget’s key-mappings will be inverted (such as inverting the traditional “increase” key-strokes and mouse-wheel motions to mean “decrease,” and vice-versa).
maximum	integer	Add, Set	Maximum value allowed by the widget.
minimum	integer	Add, Set	Minimum value allowed by the widget.
pageStep	integer	Add, Set	Integer value for the size of the increase or decrease the widget’s value should undergo when the user pressed page-up or page-down.
singleStep	integer	Add, Set	Integer value for the size of the increase or decrease the widget’s value should undergo when the user presses the up or down arrow (or clicks the up or down button).
sliderDown	True or False	Add, Set	Set true if the slider should act as if it is being pressed down by the mouse button.
sliderPosition	integer	Add, Set	Current position of the slider. This is a synonym for the “value” property.
tracking	True or False	Add, Set	If True, updates will be produced whenever the user drags the slider. If False, updates will be produced only when the user ends the drag. Defaults to True.
value	integer	Add, Set	The current value of the control.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

slider

A widget that the user can slide to choose any value between a minimum and maximum value.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
invertedAppearance	True or False	Add, Set	If True, the widget’s mapping will be mirrored, such that its minimum and maximum extreme values are in opposite locations. Defaults to False.
invertedControls	True or False	Add, Set	If True, the widget’s key-mappings will be inverted (such as inverting the traditional “increase” key-strokes and mouse-wheel motions to mean “decrease,” and vice-versa).
maximum	integer	Add, Set	Maximum value allowed by the widget.
minimum	integer	Add, Set	Minimum value allowed by the widget.
pageStep	integer	Add, Set	Integer value for the size of the increase or decrease the widget’s value should undergo when the user pressed page-up or page-down.
singleStep	integer	Add, Set	Integer value for the size of the increase or decrease the widget’s value should undergo when the user presses the up or down arrow (or clicks the up or down button).
sliderDown	True or False	Add, Set	Set true if the slider should act as if it is being pressed down by the mouse button.
sliderPosition	integer	Add, Set	Current position of the slider. This is a synonym for the “value” property.
tickInterval	integer	Add, Set	A value (in units, not pixels) that represents the desired distance between tick marks. If set to 0 (the default), the slider will choose either the single-step value or the page-step value as a default tickInterval.
tickPosition	TickPosition	Add, Set	Where the tick marks should be located.
tracking	True or False	Add, Set	If True, updates will be produced whenever the user drags the slider. If False, updates will be produced only when the user ends the drag. Defaults to True.

Property Name	Data Type	Usage	Description
value	integer	Add, Set	The current value of the control.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

spinbox

A widget that the user can use to edit, increment, or decrement a value between a minimum and maximum value.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
accelerated	True or False	Add, Set	True if the rate-of-change of the value should increase as the user holds down the increase/decrease button over a longer time.
buttonSymbols	ButtonSymbols	Add, Set	What symbols to show in the increment/decrement buttons.
frame	True or False	Add, Set	True if the widget should have a frame-rectangle drawn around it; False otherwise.
keyboardTracking	True or False	Add, Set	If True, value-changed updates are emitted while the user is entering a new value using the keyboard. If False, they are only emitted when the user has finished.
maximum	integer	Add, Set	Maximum value allowed by the widget.
minimum	integer	Add, Set	Minimum value allowed by the widget.
prefix	string	Add, Set	An optional string to display before the value string. For example, `prefix="$"` would be appropriate for displaying a dollar amount.
readOnly	True or False	Add, Set	True if the widget should be read-only; False if it should be editable.
singleStep	integer	Add, Set	Integer value for the size of the increase or decrease the widget’s value should undergo when the user presses the up or down arrow (or clicks the up or down button).
specialValueText	string	Add, Set	A special value to display when the spin-box’s value is equal to its minimum value. Typically used when the minimum value has a special/default meaning.

Property Name	Data Type	Usage	Description
suffix	string	Add, Set	An optional string to display after the numeric value. For example, `suffix=" km"`.
value	integer	Add, Set	The current value of the control.
wrapping	True or False	Add, Set	True if the dial should be allowed to wrap around from maximum to minimum values, and vice versa. Defaults to False.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clear()	Clears the widget’s content.
selectAll()	Select all of the widget’s contents.
stepDown()	Decrements the value by one step.
stepUp()	Increments the value by one step.

statusicon

A large widget that indicates the current state of the system (as seen in the upper right corner of the System Status window).

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
indent	integer	Add, Set	The label’s text-indent width, in pixels. Defaults to -1, meaning that an appropriate indent will be calculated algorithmically.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
margin	integer	Add, Set	The distance (in pixels) between the innermost pixel of the frame and the outermost pixel of the contents. Defaults to 0.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
openExternalLinks	True or False	Add, Set	True if the HTML editor should open the default web browser to show links the user clicked on; False if the linked documents should appear in the widget itself.
pixmap	string (filename)	Add, Set	Name of an image file in the Support Files window that should be loaded and displayed in this widget.
scaledContents	True or False	Add, Set	True if the label’s contents should be rescaled as the label is resized; False if they should be cropped instead.
text	string	Add, Set	Specifies the text to be displayed by the widget.
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clear()	Clears the widget’s content.

statuslist

A list of modules and their current status data (as seen in the System Status window).

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
alternatingRowColors	True or False	Add, Set	True for alternating colors, False otherwise
autoScroll	True or False	Add, Set	True if the table should auto-scroll to assist in drag-and-drop operations. Only works if drops are enabled for the table.
autoScrollMargin	integer	Add, Set	How close (in pixels) the mouse pointer needs to be to the margin of the table for auto-scroll to be initiated.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
horizontalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll horizontally.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
selectionBehavior	SelectionBehavior	Add, Set	Governs how items may be selected.
selectionMode	SelectionMode	Add, Set	Governs how (and if) multiple items may be selected.
verticalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll vertically.
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
BeginEditWithCellTextSelected()	Force the cell with the current focus into edit-mode, with the cell’s current contents selected.
RemoveSelected()	Remove any selected items from the list
clear()	Clears the widget’s content.
clearSelection()	De-selects any items that the widget may currently have selected.
hideColumn(int)	Hides the specified column
reset()	Resets the view to its default state. (Use with caution!)
resizeColumnToContents(int)	Resizes the nth column to fit the column’s current contents.
scrollToBottom()	Scrolls to the bottom of the view.
scrollToTop()	Scrolls to the top of the view.
selectAll()	Select all of the widget’s contents.
showColumn(int)	Shows the specified column, if it was previously hidden.
sortByColumn(int)	Sorts the table according to the data in the specified column

stringdialog

An input dialog that presents a text message to the user, and allows the user to enter a string value in response. Single use only (it deletes itself when the user dismisses the dialog).

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
instructions	string	Add, Set	Text telling the user what string should be entered into the dialog’s line-edit.
modal	True or False	Add, Set	True if this dialog should prevent user interaction with other GUI elements until it is closed; False otherwise. Defaults to False.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
accept()	Accept the dialog (as if the user had clicked the Okay or Accept button).
done(int)	Dismisses the dialog with the specified result-code.
open()	Tells the file dialog to open the selected file (as if the user had clicked the Open button).
reject()	Dismisses the dialog, as if the user had clicked the Cancel button.

subcueeditor

A widget that can be used to edit a specified subcue.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
subcueID	integer	Add, Set	ID of the subcue the editor should be viewing/editing.
subcueIDPath	string	Add, Set	A string indicating the ID of the subcue to view/edit, as well as contextual IDs. Only necessary when you need COW-enabled subcue editing, otherwise you can set the subcueID property instead. Supported string formats are `"subcueID"`, `"subcueID,cueID,subcueEntryID"`, or `"subcueID,cueID,subcueEntryID,cueListID,cueEntryID"`.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

subcueentrytable

A table that shows the list of subcue entries within a cue.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
alternatingRowColors	True or False	Add, Set	True for alternating colors, False otherwise
autoScroll	True or False	Add, Set	True if the table should auto-scroll to assist in drag-and-drop operations. Only works if drops are enabled for the table.
autoScrollMargin	integer	Add, Set	How close (in pixels) the mouse pointer needs to be to the margin of the table for auto-scroll to be initiated.
cueID	integer	Add, Set	ID of the Cue whose contents this table should display.
cueIDPath	string	Add, Set	A string indicating the ID of the cue to view/edit, as well as contextual IDs. Only necessary when you need COW-enabled subcue editing, otherwise you can set the cueID property instead. Supported string formats are “`cueID`”, or “`cueID,cueListID,cueEntryID`”.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
horizontalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll horizontally.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.

Property Name	Data Type	Usage	Description
selectionBehavior	SelectionBehavior	Add, Set	Governs how items may be selected.
selectionMode	SelectionMode	Add, Set	Governs how (and if) multiple items may be selected.
showGrid	True or False	Add, Set	True if the table's between-cells grid should be visible; False if it should not be. Defaults to True.
verticalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll vertically.
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see "widget" on page 40), plus:

Method Name	Description
CaptureDiffsRequested()	Request that a capture-differences action take place.
UpdateItemsRequested()	Requests that an update-items action take place.
clearSelection()	De-selects any items that the widget may currently have selected.
hideColumn(int)	Hides the specified column
hideRow(int)	Hides the specified row
reset()	Resets the view to its default state. (Use with caution!)
resizeColumnToContents(int)	Resizes the nth column to fit the column's current contents.
resizeColumnsToContents()	Resizes all columns to fit their current contents.
resizeRowToContents(int)	Resizes the nth row to fit the row's current contents.
resizeRowsToContents()	Resizes all rows to fit their current contents.
scrollToBottom()	Scrolls to the bottom of the view.
scrollToTop()	Scrolls to the top of the view.
selectAll()	Select all of the widget's contents.
showColumn(int)	Shows the specified column, if it was previously hidden.
showRow(int)	Shows the specified row, if it was previously hidden.

Method Name	Description
sortByColumn(int)	Sorts the table according to the data in the specified column

subcuelibrarytable

A table that shows the list of subcues in the project.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
alternatingRowColors	True or False	Add, Set	True for alternating colors, False otherwise
autoScroll	True or False	Add, Set	True if the table should auto-scroll to assist in drag-and-drop operations. Only works if drops are enabled for the table.
autoScrollMargin	integer	Add, Set	How close (in pixels) the mouse pointer needs to be to the margin of the table for auto-scroll to be initiated.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
horizontalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll horizontally.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
selectionBehavior	SelectionBehavior	Add, Set	Governs how items may be selected.
selectionMode	SelectionMode	Add, Set	Governs how (and if) multiple items may be selected.
showGrid	True or False	Add, Set	True if the table’s between-cells grid should be visible; False if it should not be. Defaults to True.
verticalScrollMode	ScrollMode	Add, Set	Sets the granularity with which the table should scroll vertically.
wordWrap	True or False	Add, Set	True if word wrap should be enabled; False to leave it disabled.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
clearSelection()	De-selects any items that the widget may currently have selected.
hideColumn(int)	Hides the specified column
hideRow(int)	Hides the specified row
reset()	Resets the view to its default state. (Use with caution!)
resizeColumnToContents(int)	Resizes the nth column to fit the column’s current contents.
resizeColumnsToContents()	Resizes all columns to fit their current contents.
resizeRowToContents(int)	Resizes the nth row to fit the row’s current contents.
resizeRowsToContents()	Resizes all rows to fit their current contents.
scrollToBottom()	Scrolls to the bottom of the view.
scrollToTop()	Scrolls to the top of the view.
selectAll()	Select all of the widget’s contents.
showColumn(int)	Shows the specified column, if it was previously hidden.
showRow(int)	Shows the specified row, if it was previously hidden.
sortByColumn(int)	Sorts the table according to the data in the specified column

tabbar

A widget that contains a series of tabs, of which the user can select one.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
currentIndex	integer	Add, Set	The current index that the widget is set to. Valid values range from 0 to (numberOfItems-1).
documentMode	True or False	Add, Set	True if the tabs should be rendered in document style; False (the default) if they should be rendered in the normal style.
drawBase	True or False	Add, Set	True if the tab bar should draw its base. False if there should be no base drawn.
expanding	True or False	Add, Set	True if the tab bar should expand the width of its tabs to fill up the extra space; False if they should only be large enough to fit their contents.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
movable	True or False	Add, Set	True if the user should be allowed to move the tabs within the tab bar; False (the default) otherwise.
shape	Shape	Add, Set	Selects various tab shapes.
tabsClosable	True or False	Add, Set	True if the tabs should contain close-tab buttons, False if they should not. Defaults to False.
usesScrollButtons	True or False	Add, Set	True if the tabs-strip should allow scrolling if there are too many tabs to display at once. Defaults to True.

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

textbrowser

A multi-line rich-text browser, that supports hypertext navigation.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
acceptRichText	True or False	Add, Set	Specifies whether the widget should accept rich-text insertion by the user. If set False, only plain text will be inserted. Defaults to True.
autoFormatting	AutoFormat-ting	Add, Set	What sort of automatic formatting should be enabled to help the user enter structured text.
cursorWidth	integer	Add, Set	Width of the text-editing cursor, in pixels.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
html	string	Add, Set	A string representing simple HTML source code to display.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
lineWrapColumnOr-Width	integer	Add, Set	The position (in pixels or columns, depending on the lineWrapMode property) where text will be wrapped. Defaults to 0.
lineWrapMode	LineWrapMode	Add, Set	Specifies how per-line text-wrapping should be han-dled.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
openExternalLinks	True or False	Add, Set	True if the HTML editor should open the default web browser to show links the user clicked on; False if the linked documents should appear in the widget itself.
openLinks	True or False	Add, Set	True if the HTML editor should allow the user to click links to view other HTML content; False if the links should be non-clickable.

Property Name	Data Type	Usage	Description
overwriteMode	True or False	Add, Set	If True, text entered by the user will overwrite any existing text at the cursor location. If False, new text will be inserted at the cursor location.
plainText	string	Add, Set	Plain text to display in this widget.
readOnly	True or False	Add, Set	True if the widget should be read-only; False if it should be editable.
source	string (URL)	Add, Set	Specifies the URL (or file name) of the document this widget should display.
tabChangesFocus	True or False	Add, Set	If True, the tab key will cause a focus-change to another widget. If false, the tab key will enter a tab character into the text. Defaults to False.
tabStopWidth	integer	Add, Set	Width of a tab character, in pixels.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
append(QString)	Appends the specified string to the widget’s content.
backward()	Moves back to the previous HTML document (as if the user had clicked to the ”Go Back” button in a web browser).
clear()	Clears the widget’s content.
copy()	Copies the selected contents of the widget into the global clipboard.
cut()	Cuts the selected contents of the widget into the global clipboard.
forward()	Returns `forwardback` to the next HTML document (as if the user had clicked to the “Go Forward” button in a web browser).
home()	Moves to the first document in the browser’s document-history.
insertHtml(QString)	Inserts the specified HTML into the document.
insertPlainText(QString)	Inserts the specified plain text into the document.
paste()	Pastes the clipboard’s current contents into the widget.
reload()	Re-loads the current document, if possible.

Method Name	Description
scrollToAnchor(QString)	Scrolls the view so that the specified HTML #anchor tag is visible.
selectAll()	Select all of the widget's contents.
zoomIn()	Zooms in to the displayed document, by resizing the fonts larger.
zoomIn(int)	Zooms in to the displayed document, by resizing the fonts larger by the specified amount.
zoomOut()	Zooms out from the displayed document, by resizing the fonts smaller.
zoomOut(int)	Zooms out from the display document, by resizing the fonts smaller by the specified amount.

textedit

A multi-line text-editor/text-viewer widget, that supports display and editing of plain text and rich text.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
acceptRichText	True or False	Add, Set	Specifies whether the widget should accept richtext insertion by the user. If set False, only plain text will be inserted. Defaults to True.
autoFormatting	AutoFormat-ting	Add, Set	What sort of automatic formatting should be enabled to help the user enter structured text.
cursorWidth	integer	Add, Set	Width of the text-editing cursor, in pixels.
frameRect	(left,top,width, height)	Add, Set	The rectangle that the widget’s frame-rectangle is drawn at.
frameShadow	Shadow	Add, Set	The shadowing style the frame’s rectangle will be drawn in.
frameShape	Shape	Add, Set	The shape of the frame.
html	string	Add, Set	A string representing simple HTML source code to display.
lineWidth	integer	Add, Set	The width of the frame’s rectangle.
lineWrapColumnOr-Width	integer	Add, Set	The position (in pixels or columns, depending on the lineWrapMode property) where text will be wrapped. Defaults to 0.
lineWrapMode	LineWrapMode	Add, Set	Specifies how per-line text-wrapping should be han-dled.
midLineWidth	integer	Add, Set	The width of the frame’s mid-line.
overwriteMode	True or False	Add, Set	If True, text entered by the user will overwrite any existing text at the cursor location. If False, new text will be inserted at the cursor location.
plainText	string	Add, Set	Plain text to display in this widget.

Property Name	Data Type	Usage	Description
readOnly	True or False	Add, Set	True if the widget should be read-only; False if it should be editable.
tabChangesFocus	True or False	Add, Set	If True, the tab key will cause a focus-change to another widget. If false, the tab key will enter a tab character into the text. Defaults to False.
tabStopWidth	integer	Add, Set	Width of a tab character, in pixels.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
append(QString)	Appends the specified string to the widget’s content.
clear()	Clears the widget’s content.
copy()	Copies the selected contents of the widget into the global clipboard.
cut()	Cuts the selected contents of the widget into the global clipboard.
insertHtml(QString)	Inserts the specified HTML into the document.
insertPlainText(QString)	Inserts the specified plain text into the document.
paste()	Pastes the clipboard’s current contents into the widget.
scrollToAnchor(QString)	Scrolls the view so that the specified HTML #anchor tag is visible.
selectAll()	Select all of the widget’s contents.
zoomIn()	Zooms in to the displayed document, by resizing the fonts larger.
zoomIn(int)	Zooms in to the displayed document, by resizing the fonts larger by the specified amount.
zoomOut()	Zooms out from the displayed document, by resizing the fonts smaller.
zoomOut(int)	Zooms out from the display document, by resizing the fonts smaller by the specified amount.

timeleftbutton

A momentary button whose label updates itself to the number of seconds left in the current cue recall, if any. (As seen in the GO button in the Transport window)

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoRaise	True or False	Add, Set	If True, the button will be appear to be raised (via beveling) only when the mouse is hovered over it. Defaults to False.
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if autoRepeat is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if autoRepeat is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (filename)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off

toolbutton

A minimal button that takes up a bit less space than a pushbutton.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
autoRaise	True or False	Add, Set	If True, the button will be appear to be raised (via beveling) only when the mouse is hovered over it. Defaults to False.
autoRepeat	True or False	Add, Set	If True, then holding down the mouse button on this widget will cause it to act like it is being clicked at regular intervals. Defaults to False.
autoRepeatDelay	integer	Add, Set	The initial delay (in milliseconds) before auto-repeat kicks in, if `autoRepeat` is enabled.
autoRepeatInterval	integer	Add, Set	How often (in milliseconds) auto-repeat pseudo-clicks should occur, if `autoRepeat` is enabled.
checkable	True or False	Add, Set	True if the specified widget should be check-able (gets a check mark when the user clicks on it); False if not.
checked	True or False	Add, Set	True if the specified widget should have a check-mark right now; False if it should not.
down	True or False	Add, Set	Sets whether the button is currently in its pressed-down state (as if the user was holding down the mouse button on it).
icon	string (filename)	Add, Set	File name of an image file (in the Support Files window) that should be used as the icon for this widget.
iconSize	(width,height)	Add, Set	The desired size (in pixels) for specified icons.
text	string	Add, Set	Specifies the text to be displayed by the widget.

Methods

All of the standard widget methods (see “widget” on page 40), plus:

Method Name	Description
animateClick()	Acts as if the user has clicked on the button.
animateClick(int)	Acts as if the user has clicked on the button and held it down for the specified number of milliseconds.
click()	Acts as if the user has clicked on the button without displaying any visible change. Can be used in conjunction with `animateClick()`.
toggle()	Toggles the checkbox on or off

transportview

Shows a cue list with yellow and green bars, as seen in the Transport Window.

Properties

All of the standard widget properties (see “widget” on page 40), plus:

Property Name	Data Type	Usage	Description
primaryFirstVisibleItem	integer	Add, Set	First index to display in a paged display, along the primary axis
primaryMaxVisibleItems	integer	Add, Set	Maximum allowable number of items that should be currently displayed along the primary axis
primaryMinVisibleItems	integer	Add, Set	Minimum allowable number of items that should be currently displayed along the primary axis
primaryNumVisibleItems	integer	Add, Set	Number of items that should be currently displayed along the primary axis
secondaryFirstVisibleItem	integer	Add, Set	First index to display in a paged display, along the secondary axis
secondaryMaxVisibleItems	integer	Add, Set	Maximum allowable number of items that should be currently displayed along the secondary axis
secondaryMinVisibleItems	integer	Add, Set	Minimum allowable number of items that should be currently displayed along the secondary axis
secondaryNumVisibleItems	integer	Add, Set	Number of items that should be currently displayed along the secondary axis

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

vrasdelaysgraph

A graph of the VRAS Delays for a given VRAS unit.

Properties

This widget type has only the standard widget properties (see “widget” on page 40).

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

vraserdampinggraph

A graph of the VRAS Early Reflection Damping for a given unit.

Properties

This widget type has only the standard widget properties (see “widget” on page 40).

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

vrasgraph

A graph of the VRAS transform for a given VRAS unit.

Properties

This widget type has only the standard widget properties (see “widget” on page 40).

Methods

This widget type has only the standard widget methods (see “widget” on page 40).

Meyer Sound Laboratories Inc.
2832 San Pablo Ave.
Berkeley, CA 94702

www.meyersound.com
+1 510 486.1166

CueStation Client-Side Python User Guide, PN 05.176.131.02 A