Open topic with navigation

You are here: Standards and Definitions > Reserved Functions

Reserved Functions

Following are the reserved functions within plugins. While not all of these are reserved in the traditional sense, many of the listed Design Time functions are required for the plugin to compile and are used within the Core OS Platform. If a function is not required for compile, it is noted as *optional*.

For a list of reserved control names, see Reserved Control Names.

> **Note:** Lua is case-senstive. For example, "string" and "String" are not the same.

Design Time Functions

GetProperties()

GetColor(props)

GetPrettyName(props)

RectifyProperties(props)

GetPages(props)

Optional. Used when a plugin will have multiple pages of user interface. Returns the table of page name objects. Each page name object must contain a “name” property for each desired page. Values defined by properties of the plugin can be accessed inside this function via a table which is passed to this function as an argument.

Property	Required?	Description
name	Yes	Defines the name of the page you want to add to the plugin.

> **Tip:** Pages are ordered as you define them in the table returned by this function.

This example defines a global list of page names and adds them to the plugin using the GetPages() function:

```
PageNames = { "Control", "Setup" }  --List the pages within the plugin
function GetPages(props)
  local pages = {}
  for ix,name in ipairs(PageNames) do
    table.insert(pages, {name = PageNames[ix]})
```

```
  end
  return pages
end
```

> **Note:** PageNames is defined outside the GetPages() function to allow access elsewhere in the plugin. PageNames can be modified within the Get Pages() function to add pages based on property settings. However, even though PageNames appears to be global, it is merely a constant when viewed from outside the function. Therefore, changes made here do not carry over to other UI functions.

GetControls(props)

Return a table containing the control objects used within the plugin. Controls are persistent data objects stored within the design. They can be bound to user interface components and pins by the plugin. Values defined by properties of the plugin can be accessed inside this function via a table that is passed to this function as an argument.

Name	Type	Required?	Description
Name	String	Yes	Name of the control.
ControlType	String	Yes	Defines the type of control. Options: "Button" "Knob" "Indicator" "Text"
DefaultValue	See description	No	Defines a default value for the control the first time the plugin code is compiled in a design. Type could be a Boolean, value, or string depending on the control's value.
UserPin	Boolean	No	Default is false. If true, pin will be available under "Control Pins" in the "Properties" pane of QDS.
PinStyle	String	No	Defines the style of pin when exposed. If not defined, and "UserPin" is true, the control pin will always be visible when the component is in the schematic page. Options: "Input" "Output" "Both"

			"None"
Count	Integer	No	Default is 1. Defines the number of controls created with these properties. If this value is greater than one, when accessed in the run-time code, the controls are created as an array.

> **Note:** There are no specific properties for a "Text" control. You can define whether a text control is a text box, combo box, or list box in GetControlLayout().

Button-Specific Properties

Name	Type	Required?	Description
ButtonType	String	Yes	Defines the control behavior of the button. Options: "Toggle" "Momentary" "Trigger" "StateTrigger" - For more information about StateTriggers read the help topic here.
Icon	String	No	Defines an icon for the control. The icon can be from QDS (e.g., "skull") or from a local file. To learn how the plugin compiler can encode a local image, see Plugin Compiler.
IconType	String	Yes (if Icon is defined)	Defines the format of the image file. The default value is 'Icon', and the 'Image' supports PNG (with alpha-channel) and JPG. Options: SVG, Image, Icon.
Max	Integer	Yes (if ButtonType is StateTrigger)	Defines the maximum value for a StateTrigger.
Min	Integer	Yes (if ButtonType is StateTrigger)	Defines the minimum value for a StateTrigger.

Knob-Specific Properties

Name	Type	Required?	Description
			Defines the units of the control. Options:

ControlUnit	String	Yes	"dB" "Hz" "Float" "Integer" "Pan" "Percent" "Position" "Seconds"
Max	Integer	See Below	Defines the maximum value of the knob. Must be greater than the "Min" value. Default value and range limits are determined by the particular "ControlUnit" selected. See Knob-Specific Ranges for details.
Min	Integer	See Below	Defines the minimum value of the knob. Must be less than the "Max" value. Default value and range limits are determined by the particular "ControlUnit" selected. See Knob-Specific Ranges for details.

Knob-Specific Ranges

Control Unit	Lower Limit	Upper Limit	Default Min	Default Max	Min/Max Required
dB	-100	20	-100	20	Yes
Hz	20	20000	20	20000	Yes
Float	-1,000,000,000	1,000,000,000	0	100	Yes
Integer	-999,999,999	999,999,999	1	100	Yes
Pan	-1	1	-1	1	No
Percent	0	100	0	100	Yes
Position	0	1	0	1	No
Seconds	0	87400	0	1	Yes

Indicator-Specific Properties

Name	Type	Required?	Description
			Defines the type of indicator to be displayed. Options: "Led"

IndicatorType	String	Yes	"Meter" "Text" "Status"

```
function GetControls(props)
  --[[
  This function uses a local 'controls' table. This should not be confused with the "Cont
  --]]
  local controls = {}
  --single Control
  table.insert(controls,{
    Name = "TextBox",
    ControlType = "Text",
    UserPin = false,
    Count = 1
  })
  --multiple controls (note Count value)
  table.insert(controls,{
    Name = "LED Indicator",
    ControlType = "Indicator",
    IndicatorType = "Led",
    UserPin = true,
    PinStyle = "Output",
    Count = 5
  })
  return controls
end
```

GetControlLayout(props)

Returns the layout and graphics tables that define the UI view of the plugin.

The layout table is comprised of objects that define how each of the controls defined in GetControls() will be shown in the plugin's UI. Each object's name must match a control defined in the GetControls() function.

If more than one of a specific control are defined, each control's name is comprised of the control name, followed by a space, and then a numerical suffix. For example, a button control defined with a name of "myBtn" and a set "Count" of 3 in GetControls() would create controls called "myBtn 1", myBtn 2" and "myBtn 3", respectively.

The graphics table is comprised of objects for each of the unbound graphics in the UI, such as labels, images, and group boxes.

Layout Table Properties

Name	Type	Required?	Description
Position	Table	Yes	Defines the position of the control. Given in the form {x,y}.

Size	Table	Yes	Defines the size of the control. Given in the form {x,y}.
Style	String	Yes	Defines how control is displayed. Options: "Fader" "Knob" "Button" "Text" "Meter" "Led" "ListBox" "ComboBox" "Media" "None" - This can be used when a control needs to be hidden but access to its control pins is still required.
ClassName	String	No	Defines the Default CSS Class Name of the control for when it is dragged onto a UCI.
Color	Table	No	Defines the color of the control. Given in the form {r,g,b,alpha}. Alpha is optional. Range: 0 - 255
TextColor	Table	No	Color of text. Given in the form {r,g,b,alpha}. Alpha is optional. Range: 0 - 255
Font	String	No	Default is "Roboto" with FontStyle="Regular". See Font and FontStyle Combinations for all valid combinations.
FontSize	Integer	No	Size of font used for text.
FontStyle	String	No	See Font and FontStyle Combinations for all valid combinations.
IsBold	Boolean	No	Sets the FontStyle to Bold.
HTextAlign	String	No	Options: "Center" "Left" "Right"

			Default is "Center"
IsReadOnly	Boolean	No	Control cannot be changed at run-time. This is good for status readouts, not necessary for indicators. (Use Status Text control without this property.)
Margin	Integer	No	Sets margin for control. Default is 0.
Padding	Integer	No	Number of pixels to pad the control graphics. Default is 1.
PrettyName	String	No	Used to create alternate names for control pins. Use "~" to create a separate sub-level.
Radius	Integer	No	Same as CornerRadius.
StrokeColor	Table	No	Color of outline. Provided in the form {r,g,b,alpha}. Alpha is optional. Default is black {0,0,0}. Range: 0 - 255
StrokeWidth	Integer	No	Thickness of outline. Default is 1.
VTextAlign	String	No	Options: "Center" "Top" "Bottom" Default is "Center"
ZOrder	Signed Integer	No	Sets the position of the control in the vertical plane. Range is -2147483648 to 2147483647. See notes on Object Layering below.

Button-Only Properties

Name	Type	Required?	Description
ButtonStyle	String	Yes	Options: "Toggle" "Momentary" "Trigger" "StateTrigger" "On"

			"Off" "Custom" For a "string" type button, use "Custom'".
ButtonVisualStyle	String	No	Options: "Flat" "Gloss" Default is "Gloss".
CornerRadius	Integer	No	Sets radius of corners
Radius	Integer	No	Same as CornerRadius.
CustomButtonUp	String	No	Defines the string of a Custom button when in the up position.
CustomButtonDown	String	No	Defines the string of a Custom button when in the down position.
Legend	String	No	Defines a legend for the button.
OffColor	Table	No	Used only when UnlinkOffColor is true. Sets the color of the button when it's off. Provided in the form {r,g,b,alpha}. Alpha is optional. Range: 0 - 255
UnlinkOffColor	Boolean	No	Allow a button to show two completely different colors for on and off states.
IconColor	Table	No	Sets the color of the buttons Icon. Provided in the form {r,g,b,alpha}. Alpha is optional. Range: 0 - 255
WordWrap	Boolean	No	Defines whether the legend text should wrap.

> **Note:** The "Off", "On" and "Custom" options are the equivalent of the "Off", "On" and "String" push-actions as described in Q-SYS Help.

Fader-Only Properties

Name	Type	Required?	Description
			Add a text box to show the current control's value.

ShowTextbox	Boolean	No	Default is "false".

Meter-Only Properties

Name	Type	Required?	Description
CornerRadius	Integer	No	Sets radius of corners
Radius	Integer	No	Same as CornerRadius.
BackgroundColor	Table	No	Sets background color for the meter. Provided in the form {r,g,b,alpha}. Alpha is optional. Range: 0 - 255
MeterStyle	String	Yes	Options: "Level" "Reduction" "Gain" "Standard"
ShowTextbox	Boolean	No	Add a text box to show the current control's value. Default is "true".

Text-Only Properties

Name	Type	Required?	Description
CornerRadius	Integer	No	Sets radius of corners
Radius	Integer	No	Same as CornerRadius
TextBoxStyle	String	No	Options: "Normal" "Meter" "NoBackground" Default is "Normal"
WordWrap	Boolean	No	Defines whether the legend text should wrap

Graphics Table Properties

Name	Type	Required?	Description
Position	Table	Yes	Defines the position of the control. Given in the form {x,y}.
Size	Table	Yes	Defines the size of the control. Given in the form {x,y}.
Type	String	Yes	Options: "Label" - Fixed text label "GroupBox" - Outline to enclose controls with or without title "Header" - Single line with header text "Image" - An image except SVG (see SVG type) "Svg" - Image
ZOrder	Signed Integer	No	Sets the position of the graphic in the vertical plane. Range is -2147483648 to 2147483647. See notes on Object Layering below.

GroupBox-Only Properties

Name	Type	Required?	Description
CornerRadius	Integer	No	Sets radius of corners on the group box.
Radius	Integer	No	Same as CornerRadius.
Text	String	No	Default is no text shown.
Font	String	No	Default is "Roboto" with FontStyle="Regular". See Font and FontStyle Combinations for all valid combinations.
FontSize	Integer	No	Default FontSize varies by style type.
FontStyle	FontStyle	No	See Font and FontStyle Combinations for all valid combinations.
IsBold	Boolean	No	Sets the FontStyle to Bold.
HTextAlign	String	No	Options: "Center", "Left", "Right". Default is "Center".

StrokeWidth	Integer	No	Defines the width of the stroke around the control. Default is 0. Options: 0-64.
StrokeColor	Table	No	Defines the color of the stroke around the control. Default is black. Provided in the form {r,g,b,alpha}. Alpha is optional. Range: 0 - 255
Color	Table	No	Text Color provided in the form {r,g,b,alpha}. Alpha is optional. Range: 0 - 255
Fill	Table	No	Default is clear. Provided in the form {r,g,b,alpha}. Alpha is optional. Range: 0 - 255

Header-Only Properties

Name	Type	Required?	Description
Text	String	No	Default is no text shown.
Font	String	No	Default is "Roboto" with FontStyle="Regular". See Font and FontStyle Combinations for all valid combinations.
FontSize	Integer	No	Default FontSize varies by style type.
FontStyle	FontStyle	No	See Font and FontStyle Combinations for all valid combinations.
IsBold	Boolean	No	Sets the FontStyle to Bold.
HTextAlign	String	No	Options: "Center" "Left" "Right" Default is "Center"
Color	Table	No	Set the text color. Provided in the form {r,g,b,alpha}. Alpha is optional. Range: 0 - 255

Image/SVG-only Properties

Name	Type	Require?	Description
Image	String	Yes	Contains the entire Base-64-encoded JPG/PNG/SVG string for the Image or Svg types.

Label-only Properties

Name	Type	Required?	Description
Color	Table	No	Sets color of label. Provided in the form {r,g,b,alpha}. Alpha is optional. Range: 0 - 255
CornerRadius	Integer	No	Sets radius of corners on the group box.
Radius	Integer	No	Same as CornerRadius.
Margin	Integer	No	Sets margin for control. Default is 0.
Padding	Integer	No	Number of pixels to pad the border around the text. Default is 1.
Text	String	No	Default is no text shown.
Font	String	No	Default is "Roboto" with FontStyle="Regular". See Font and FontStyle Combinations for all valid combinations.
Fill	Table	No	Default is clear. Provided in the form {r,g,b,aplha}. Alpha is optional.
FontSize	Integer	No	Default FontSize varies by style type.
FontStyle	FontStyle	No	See Font and FontStyle Combinations for all valid combinations.
IsBold	Boolean	No	Sets the FontStyle to Bold.
HTextAlign	String	No	Options: "Center", "Left", "Right". Default is "Center".
VTextAlign	String	No	Options: "Center", "Top", "Bottom". Default is "Center"
StrokeWidth	Integer	No	Defines the width of the stroke around the control. Default is 1. Options: 0-64
StrokeColor	Table	No	Defines the color of the stroke around the control. Default is black. Provided in the form {r,g,b,alpha}. Alpha is optional. Range: 0 - 255

Font and FontStyle Combinations

> **Note:** FontStyle capitalization varies between fonts.

Font	Link to Example	FontStyle
Adamina	Adamina	Regular
Droid Sans	No Link	Regular, Bold
Lato	Lato	Light, Light Italic, Regular, Italic, Bold, Bold Italic, Black, Black Italic
Montserrat	Montserrat	Thin, Thin Italic, ExtraLight, ExtraLight Italic, Light, Light Italic, Regular, Italic, Medium, Medium Italic, SemiBold, SemiBold Italic, Bold, Bold Italic, ExtraBold, ExtraBold Italic, Black, Black Italic
Noto Serif	Noto Serif	Regular, Italic, Bold, BoldItalic
Open Sans	Open Sans	Light, Light Italic, Regular, Italic, Semibold, Semibold Italic, Bold, Bold Italic, Extrabold, Extrabold Italic
Poppins	Poppins	Light, Regular, Medium, SemiBold, Bold
Roboto	Roboto	Thin, Thin Italic, Light, Light Italic, Regular, Italic, Medium, Medium Italic, Bold, Bold Italic, Black, Black Italic
Roboto Mono	Roboto Mono	Thin, Thin Italic, Light, Light Italic, Regular, Italic, Medium, Medium Italic, Bold, Bold Italic
Roboto Slab	Roboto Slab	Thin, Light, Regular, Bold
Slabo 27px	Slabo 27px	Regular

```
function GetControlLayout(props)
  local layout = {}
  local graphics = {}

  --Example of a single textbox control
  layout["TextBox"]={
    Style="Text",
    IsReadOnly=true,
    Position={0,0},
    Size={100,16},
    TextColor={255,255,255},
    TextBoxStyle="NoBackground",
    FontSize=9,
    HTextAlign="Left",
    VTextAlign="Center",
    Font = "Montserrat",
    FontStyle = "SemiBold",
  }

  --Example of defining multiple LED indicators
  for idx = 1,5 do
```

```
    layout["LEDIndicator "..i]={
      PrettyName="LED~State",
      Style="LED",
      Color={0,255,0},
      OffColor={0,0,0},
      UnlinkOffColor=true,
      Position={0,20},
      Size={16,16},
      CornerRadius=8
    }
  end

  --Plugin Background Color - Note: As "Fill" and "StrokeColor" are the same color so the
  table.insert(graphics,{
    Type="GroupBox",
    Position={0,0},
    Size={200,200},
    Fill={0,0,0},
    CornerRadius=0,
    StrokeColor={0,0,0},
    StrokeWidth=1
  })

  --Label
  table.insert(graphics,{
    Type="Label",
    Text="This is a label",
    Position={0,40},
    Size={100,14},
    Color={255,255,255},
    FontSize=9,
    HTextAlign="Left"
  })

  return layout,graphics
end
```

Object Layering

QDS normally handles how controls and graphics objects are overlaid over one another (the z-order). When building a plugin, `ZOrder` can be specified as a signed integer for controls and graphics to manually set where on the vertical plane an object resides. The higher the value, the closer to the front the object will be. The "stacking" of items using `ZOrder` applies equally to graphic elements and controls as they exist in the same layering scheme. This allows graphics to be placed above controls if required. In other words, what you do to any one graphic or control potentially has effects on any object on that page (graphic or control).

The `ZOrder` key can be added to the layout or graphic control definition at any point, as shown in the following example for a TextBox control.

```
layout["TextBox"]={
  Style="Text",
  IsReadOnly=true,
  Position={0,0},
  Size={100,16},
  ZOrder=23,
  TextColor={255,255,255},
  TextBoxStyle="NoBackground",
  FontSize=9,
  HTextAlign="Left",
```

```
  VTextAlign="Center"
}
```

> **Note:** If `ZOrder` is used to manually set the order on ANY control or graphic, it is recommended that it be used on all controls or graphics. Otherwise, strange effects may occur.

Using Multiple Pages

For plugins that use multiple pages, the control layout needs to be split into two distinct sections:

Controls and graphics that are common and appear on all pages.

Controls and graphics that only appear on a specific page.

Those that are common can be added as normal. However, an additional condition needs to be included to separate which controls are shown on which page.

This condition can use the numeric index value of the page:

```
-- Common controls and graphics go here...
if props['page_index'].Value == 1 then
  -- Controls and graphics which appear on first page
elseif props['page_index'].Value == 2 then
  -- Controls and graphics which should appear on second page and so on
end
```

Alternatively, you can use the page's actual name instead:

```
local CurrentPage = pagenames[props["page_index"].Value]
--Common controls and graphics go here
if CurrentPage == "Page 1" then
  -- Controls and graphics which appear on "Page 1" here
elseif CurrentPage == "Page 2" then
  -- Controls and graphics which appear on "Page 2" here and so on...
end
```

> **Tip:** A typical use case in a multi-page plugin is to alter the pages the plugin shows based on what property values are set. An array is used to define the default pages and then, in GetProperties(), additional pages can be added based on the property settings. However, when GetControls() or Get ControlLayout() are called, the page name array will revert back to just the default pages. The changes made in GetProperties() will not carry over. As the page name array may be needed in GetControls() or GetControlsLayout(), the full array will need to be recreated from scratch from within these functions. The way to do that is to create a

> separate helper function that creates the full page name array based on the properties and call that helper function from within GetControls() or GetControlLayout() to recreate the page name array that is required.

GetComponents(props)

GetPins(props)

GetWiring(props)

Runtime

This information is provided to further increase understanding of QSC-authored plugins. These functions are not required to be added to a user-authored plugin. The following are common functions typically found in QSC-authored plugins. For a working example of these functions, see TCPSocket.

SetupDebugPrint()

Send(cmd)

ClearVariables()

Connect()

Disconnected()

GetDeviceInfo()

PollDevice()

ParseResponse()

Initialization()