AddonsForLife API documentation

The following base files hold the API:

|  |  |
| --- | --- |
| error.lua | Loaded first – vital error handling routines.  Not public, overrides error(). |
| base.lua | C#/Lua communication and basic UI structures.  Not public. This file hides C# functions after hooking them. |
| tags.lua | Tag system, not used right now. |
| UI.lua | Compound UI structures, most classes here are public. |

Note that when a function has the form parent.func(self, …), it is generally expected to be used using the parent:func(…) syntax.

# error.lua:

Identical to equivalent Lua functions, but provide enhanced functionality.

## print(…)

Now prints almost any type of data, showing the values of booleans, the name of userdatas, and the top level structure of tables.

## error(…)

Gives a more complete stack, routed through C#.

# base.lua:

NOTE: UI elements may sometimes be referred to by a number, and sometimes by table.

NOTE: This file seals off direct access to the C# functions. Any Lua code written after the end of this file must call existing Lua code.

## dprint(…)

Prints messages if debugging is enabled.

## registerEvent(ID, name, method)

Registers object “ID” with event “name.” “method” is called when the event fires.

Some events originate from C#.

## setGridPosition

No longer used.

## CreateFrame()

Creates a new window, with the standard borders used by Windows.

Returns a table representing the Frame, with the following helper methods:

### SetParent(self, parent)

Sets the parent of the frame (simply calls AddChild).

### AddChild(self, child)

Sets “child” to be the child of the current frame.

### SetPosition(self, x, y)

Sets the position of the frame relative to the desktop.

### getIndex(self)

Gets the index of the current frame. Note that the index cannot be changed: It’s required to communicate with C#. I recommend creating your own indexing system if this one is unsuitable for your needs.

### setTitle(self, text)

Sets the title of the frame.

## Events available for Frame:

#### OnClick

Fires when the mouse button is released and the mouse cursor is above the frame.

## CreatePane(parent)

Note: All panes have a parent. For top level panes, you can use a Frame or a Region to hold the pane.

This is one of the fundamental building blocks of creating a visible UI – most complex UI elements use this as a base. It can have a background and listens for events.

If parent is nil, then the Region representing the primary monitor is used.

Returns a table representing the Frame, with the following helper methods:

### SetParent(self, parent)

Sets the parent of the pane (simply calls AddChild).

### AddChild(self, child)

Sets “child” to be the child of the current frame.

### RemoveChild(self, child)

Removes the child indicated by “child.”

### RemoveAllChildren(self)

Removes all children from the pane.

### SetPosition(self, x, y)

Sets the position of the pane relative to its parent.

### SetSize(self, x, y)

Sets the size of the pane relative to its parent.

### SetBackground(self, file)

Sets the background of the pane using “file” as the file name. The image is stretched to fill the pane (more options may be available in the future).

### SetBorderedBackground(self, file, width, height)

Creates a border style background from the file:

* The image is split into parts to form a border effect.
* Width sets how thick the left and right edges are.
* Height sets how thick the top and bottom edges are.
* The edges and center are scaled appropriately as the pane is resized. The corners stay at their original size.

### getIndex(self)

Similar to the Frame index. Not allowed to be changed.

### registerEvent(self, name, method)

Registers an event with the current pane. “name” is the name of the event to register, “method” is called when the event fires.

### setPosition(self, x, y)

Sets the current position of the pane relative to its parent. (oops, this is a redundancy!)

### setTitle(self, text)

Will be available to set a title (to be shown at the top of the pane, probably with its own background) in the future.

## Events available to Panes:

#### OnClick

Fires when the mouse button is released.

#### OnMouseEnter

Fires when the cursor enters the visible area of the pane.

#### OnMouseLeave

Fires when the cursor leaves the visible area of the pane.

## CreateText(parent)

Works similarly to the Pane, except this structure can display text. This is intended to be the second fundamental building block of UIs, and gives them the ability to display text.

Right now, this is a fairly heavyweight element, inheriting a lot from Pane.

NOTE: Text that is too large to fill the control is aligned to the left and clipped.

Returns a table representing the Text, with the following helper methods:

### getIndex(self)

Similar to the Frame index. Not allowed to be changed.

### SetText(self, text)

Sets the text that is visible in the Text element.

### registerEvent(self, name, method)

Registers an event with the current text. “name” is the name of the event to register, “method” is called when the event fires.

### setHAlign(self, text)

Sets the alignment of the text horizontally. Valid values are:

* “left” aligns the text to the left.
* “center” aligns the text to the center.
* “right” aligns the text to the right.
* “stretch” has meaning in C#, but due to the current implementation, it acts like “left.”
* Current implementation will throw an error from C# if the value is invalid.

### setVAlign(self, text)

Sets the alignment of the text vertically. Valid values are:

* “top” aligns the text to the top.
* “center” aligns the text to the center.
* “bottom” aligns the text to the bottom.
* “stretch” has meaning in C#, but due to the current implementation, it acts like “top.”
* Current implementation will throw an error from C# if the value is invalid.

### SetBackground(self, file)

Sets the background of the text using “file” as the file name. The image is stretched to fill the bounds of the text.

### SetBorderedBackground(self, file, width, height)

Creates a border style background from the file:

* The image is split into parts to form a border effect.
* Width sets how thick the left and right edges are.
* Height sets how thick the top and bottom edges are.
* The edges and center are scaled appropriately as the text is resized. The corners stay at their original size.

### SetPosition(self, x, y)

Sets the position of the text relative to its parent.

### SetSize(self, x, y)

Sets the size of the text relative to its parent.

## Events available to Text:

#### OnClick

Fires when the mouse button is released.

## CreateTextBox()

Creates a text box that has editable text.

### getIndex(self)

Similar to the Frame index. Not allowed to be changed.

### SetText(self, text)

Sets the text.

### GetText(self)

Gets the current text.

### registerEvent(self, name, method)

Registers an event with the current text box. “name” is the name of the event to register, “method” is called when the event fires.

### SetBackground(self, file)

Sets the background of the text box using “file” as the file name. The image is stretched to fill the bounds of the text.

### setHAlign(self, text)

Sets the alignment of the text horizontally. Similar to Text.

### setVAlign(self, text)

Sets the alignment of the text vertically. Similar to Text.

## Events available to Text Box:

#### OnClick

Fires when the mouse button is released.

#### OnTextChanged

Fires when the text is changed.

# Regions:

Regions are special structures that act like Panes – however, the user cannot create them. They are invisible panes that cover the entirety of each available monitor on the user’s system. One of the regions is the primary, which is the primary monitor on the system.

NOTE: Regions currently do not change if the user changes the arrangement of the monitors or sets a different primary monitor.

The functions available to regions are the same as the Panes, except the functions to resize and move them have been removed.

Regions have their own indexes, and the structure is as follows:

Region[regionIndex]=(table similar to Pane’s)

Region.primary=regionIndex

# UI.lua

This file contains compound UI structures derived from the basic structures in base.lua. There is no code that directly communicates with C# here – everything is done via Lua and previously defined functions.

## CreateListBox2(frame)

Creates a ListBox as a child of frame. Probably going to be deleted.

## CreateButton(frame)

Creates a button as a child of “frame” (which can be a Frame or a Pane or a Region).

Actually creates a Text control that has default values for the background and alignment, to make it look like a button. Use OnClick to detect mouse presses.

## CreateListBox(panelist, parent, numPanes)

Creates a ListBox using paneList to fill its own panes. Uses “parent” as the parent, and creates numPanes items in visible list.

Returns a table representing the Text, with the following helper methods:

### SetPosition(self, x, y)

Sets the position of the list box relative to its parent.

### SetSize(self, x, y)

Sets the size of the lit box relative to its parent.

### getCellWidth()

Gets the width of the individual cells.

### getCellHeight()

Gets the height of the individual cells.