LibAddonMenu-2.0 Docs

Differences between LibAddonMenu-1.0 (LAM1) and LibAddonMenu-2.0 (LAM2):

1. LAM1 was function-based for authors using it, whereas LAM2 is more table-based. Instead of calling a function for every options control you wish to create, with LAM2 you create a table that houses all of the data for your panel and another for the data of your options controls, and then pass these tables to LAM2 through only two functions (one to register your addon and create a panel, and the other to assign your options to the panel).

2. LAM2 comes with 85% less hacks around ESO's default options controls interface and templates. This makes it cleaner, leaner and smoother.

3. Getting your hands dirty with LAM1 was difficult, to say the least. It is all much easier with LAM2. Want a global reference to your control to manipulate it later? There's an optional field for that in each option's data table. (References are now optional so that they don't clutter the global namespace.) Use this reference to alter your control, call control:UpdateValue() or control:UpdateDisabled(), or add your own methods to it.

4. Going along with #3 above, there is a new control type called "custom." This allows you to basically create a container in a LAM2 addon panel where you can place any sort of custom control you wish to be in your GUI.

5. LAM1 forced you into using the LAM1 controls with a LAM1 created addon panel. With LAM2, the methods to create the controls are exposed, allowing you to use LAM2 as a factory for building options. If you want it done, you can probably do it. Use LAM2 controls without a LAM2 panel, or create a LAM2 panel but use your own controls, or use both (LAM2 panel with LAM2 controls), but create it all manually yourself, instead of letting LAM2 handle anchoring, etc. for you.

6. Four new features are available in LAM2 that were not present in LAM1:
    a) Controls now have an :UpdateValue() method. This allows you to manually change the value the control is set to.
    b) Controls may now be disabled based on the value of another control. There is also an :UpdateDisabled() method for controls.
    c) Panels will now refresh what is displayed when a setting is changed or the panel is shown if the registerForRefresh field is set to true for the panel. This picks up any changes made to the saved variables outside of the options controls.
    d) Controls may have a default value to be used if LAM2's new "Reset To Defaults" button is pressed. This is enabled if the registerForDefaults field is set to true for the panel.

7. In LAM1, a checkbox control had to toggle itself through myVar = not myVar. In LAM2, the value of the checkbox is passed through to your "set" function. This ensures your variable is always set to the correct value with the new :UpdateValue() method and panel refresh feature.

8. Controls in LAM1 were only anchored vertically, so as to mirror the default options panels. LAM2 allows you to set a control's width to "half", which allows 2 controls per line.

9. There have been concerns with the number of entries added to the game menu due to the popularity of LAM1 with addon authors. LAM2 now only creates one entry in the menu, called "Addon Settings". The main LAM2 panel houses a menu to select which registered addon's options to view - and it's alphabetized.

10. ESO has a problem with creating many controls at once, which throws an error onto confused users' screens. LAM2 holds back creation of options controls until the addon's panel is viewed for the first time. In addition to

controls not all being created at the same time, this has the added benefit of not eating up resources if a user doesn't view the options panel that session.

Getting started with LAM2 - A Beginner's Guide:

First, you will need to create a table for your panel data and another table for your options data. (These two tables may be contained in another table, however the two tables must be separate from each other when passed to LAM2.)

At the top of panel.lua (found in LibAddonMenu-2.0\controls) you will find what keys (fields) must be found in your table, what fields are optional, and what type of data may be assigned to them. Feel free to store more data in your table that you will require for your addon. This table will be stored to the panel object created for your addon in the "data" field.

Example using only the required fields:

```
local panelData = {
    type = "panel",
    name = "My Addon",
}
```

Then register your addon with LAM2 along with this table. The first arg passed to the function is a *unique* reference to your addon panel - this will be used as the panel's global name, so be specific. (As always, grab the copy of the library through LibStub first.)

```
local LAM2 = LibStub("LibAddonMenu-2.0")
LAM2:RegisterAddonPanel("MyAddonOptions", panelData)
```

Like at the top of panel.lua, the file for each control contains each required and optional field that may be set in its data table. Again, additional data may be stored in the table if required, and may be found in the data table on the control object.

Below, you will find an example creating a checkbox, dropdown and slider control. The control data tables are stored in the main options data table in index-value pairs. The index is numerical, and is the order in which your controls will be created and arranged. The value assigned to each index is the table in which the control's data is stored.

Example using only the required fields:

```
local optionsData = {
    [1] = {
        type = "checkbox",
        name = "My Checkbox",
        tooltip = "Checkbox's tooltip text.",
        getFunc = function() return true end,
        setFunc = function(value) d(value) end,
    },
    [2] = {
        type = "dropdown",
        name = "My Dropdown",
        tooltip = "Dropdown's tooltip text.",
        choices = {"table", "of", "choices"},
```

```
            getFunc = function() return "of" end,
            setFunc = function(var) print(var) end,
        },
        [3] = {
            type = "slider",
            name = "My Slider",
            tooltip = "Slider's tooltip text.",
            min = 0,
            max = 20,
            getFunc = function() return 3 end,
            setFunc = function(value) d(value) end,
        },
    }
```

You can find examples for each control type, including how to nest controls within a submenu and its data table, in exampleoptions.lua.

Register your options to your addon panel using the same unique reference you assigned it in the previous function call to LAM2.

```
LAM2:RegisterOptionControls("MyAddonOptions", optionsData)
```

LAM2 for the Experienced Author:

As mentioned in the “Differences...” section above, LAM2 offers the ability for an author to customize the options display.

1. Use a custom control to create a container for your custom widget object.

2. Set a control to “half” width to customize the anchoring of your controls. This works with any control type (except submenu), including headers and descriptions.

3. Use LAM2 to create a panel for your addon, but create your options controls and anchor them manually, however you wish.
   a) Use the LAMCreateControl methods manually
   b) Create your own custom controls

Notes:

- ✓ Don’t colorize your addon name in your panel’s “name” field. This messes up alphabetizing. Use the displayName field for any color or texture codes.

- ✓ When converting from LAM1 to LAM2, don’t forget that checkboxes pass through the value now - you should use this so that your saved variables don’t get out of sync with the display.

- ✓ Make sure the unique reference passed to LAM2:RegisterAddonPanel() is not generic and is not the same as another global variable in your addon. “MyAddon” is not unique. “SeerahsInventoryAddon” is, for example.

Exposed Methods on LAM2:

REGISTER ADDON PANEL

Registers your addon with LibAddonMenu and creates a panel

:RegisterAddonPanel(addonID, panelData)

Usage:

addonID = "string"; unique ID which will be the global name of your panel

panelData = table; data object for your panel - see controls\panel.lua

Returns:

panel = userdata; a reference to the created addon panel

REGISTER OPTION CONTROLS

Registers the options you want shown for your addon. These are stored in a table where each key-value pair is the order of the options in the panel and the data for that control, respectively. See exampleoptions.lua for an example. See controls\<widget>.lua for each widget type

:RegisterOptionControls(addonID, optionsTable)

Usage:

addonID = "string"; the same string passed to :RegisterAddonPanel

optionsTable = table; the table containing all of the options controls and their data

OPEN TO ADDON PANEL

Opens to a specific addon's option panel.

:OpenToPanel(panel)

Usage:

panel = userdata; the panel returned by the :RegisterOptionsPanel method

REGISTER WIDGET (advanced)

Each widget has its version checked before loading, so we only have the most recent one in memory.

:RegisterWidget(widgetType, widgetVersion)

Usage:

widgetType = "string"; the type of widget being registered

widgetVersion = integer; the widget's version number

CREATE CONTROL (advanced)

For manual control creation. You must handle anchoring of these controls, either to a LAM panel or a control of your choosing.

LAMCreateControl[widgetType](parent, widgetData, widgetName)

Usage:

widgetType = “string”; the type of widget to be created

parent = userdata; the panel returned by the :RegisterOptionsPanel method

widgetData = table; the table containing the widget’s data

widgetName = “string”; a *unique* global reference for your control (optional)

Returns:

widget = userdata; a reference to the control created