# Mccprefs.mui

## Super class

Group.mui

## Background

Mccprefs is the base class for all custom class configuration classes. Each configuration class (*.mcp) must be a subclass of Mccprefs.

Mccprefs does some internal house keeping stuff for configuration objects and also offers public methods that subclasses may call to improve their appearance.

## Methods

| Method | Version |
|---|---|
| MUIM_Mccprefs_ConfigToGadgets | V11 |
| MUIM_Mccprefs_GadgetsToConfig | V11 |
| MUIM_Mccprefs_RegisterGadget | V20 |

## MUIM_Mccprefs_ConfigToGadgets

### NAME

MUIM_Mccprefs_ConfigToGadgets — V11, 0x80427043

### SYNOPSIS

```
DoMethod(obj, MUIM_Mccprefs_ConfigToGadgets, Object *configdata);
```

### FUNCTION

This method will be called whenever MUI wants your custom prefs class to transfer configuration values from the configuration to the GUI.

## INPUTS

Object *configdata
: an instance of Dataspace class which carries all configuration values. Call MUIM_Dataspace_Find on this object to obtain the configration values which belong to your class.

## SEE ALSO

MUIM_Mccprefs_GadgetsToConfig, MUIM_Mccprefs_RegisterGadget

# MUIM_Mccprefs_GadgetsToConfig

## NAME

MUIM_Mccprefs_GadgetsToConfig — V11, 0x80425242

## SYNOPSIS

```
DoMethod(obj, MUIM_Mccprefs_GadgetsToConfig, Object *configdata, Object *originator);
```

## FUNCTION

This method will be called whenever MUI wants your custom prefs class to transfer configuration values from the GUI to the configuration.

## INPUTS

Object *configdata
: an instance of Dataspace class which carries all configuration values. Call MUIM_Dataspace_Add to add all configration values which belong to your class and which don't match their default values. Call MUIM_Dataspace_Remove to remove a configuration value, i.e. if it matches the default value.

## SEE ALSO

MUIM_Mccprefs_ConfigToGadgets, MUIM_Mccprefs_RegisterGadget

# MUIM_Mccprefs_RegisterGadget

## NAME

MUIM_Mccprefs_RegisterGadget — V20, 0x80424828

## SYNOPSIS

```
DoMethod(obj, MUIM_Mccprefs_RegisterGadget, Object *gadget, ULONG id, ULONG params,
CONST_STRPTR title, ULONG attr, Object *label);
```

## FUNCTION

Registers a gadget with the MUI custom class preferences system.

Preferences classes usually contain gadgets which are directly related to certain configuration options of your class, i.e. Poppen objects to adjust colors, String objects to adjust fonts, etc. These classes are directly related to a configuration ID value that you use to save their contents in the application's dataspace.

By default, MUI does not know anything about your objects and what they are doing in your preferences class. With MUIM_Mccprefs_RegisterGadget, you can tell MUI which object is related to what configuration ID value. This knowledge helps MUI to improve handling of your gadget.

Since version 20 of muimaster.library, MUI will e.g. add a default popup menu to all registered gadgets. Your object will automatically receive the same functionality as MUI's internal settings, like

"Reset to defaults", "Last Saved", "Restore", "Presets…"

Registering gadgets with MUIM_Mccprefs_RegisterGadget should be done right after the gadget was created.

There's nothing more to be done to add the extra functionality to your gadgets besides registering them with this method. MUI can then automatically perform all the actions above (defaults, last saved, etc.) by altering the configuration dataspace while using the ID value you provided as a filter.

## NOTES

Whenever the user selects something from the popup menu, your mcp class might receive the methods MUIM_Mccprefs_ConfigToGadgets and MUIM_Mccprefs_GadgetsToConfig. Anyway, you don't have to implement special treatment for popup menus there. Simply let these methods do what they usually do, i.e. setup your gadget with the configuration values from the dataspace or fill the dataspace with the contents of your gadgets.

Previous custom class examples might use the names like MUIM_Settingsgroup_ConfigToGadgets? and MUIM_Settingsgroup_GadgetsToConfig? instead of their Mccprefs counterparts. Don't be confused, both versions refer to the same values and are identical. However, _Mccprefs_ is the correct term and should be used in new code.

In the rare case that your preferences class is dynamic, i.e. that you delete/recreate configuration gadgets on the fly, you must remember to "unregister" registered gadgets before they are disposed. To unregister, call MUIM_Mccprefs_RegisterGadget with an ID parameter of 0. Usually, your gadgets won't be dynamic, so unregistering is not necessary.

## INPUTS

Object *gadget
: the object to be registered. Only objects that are directly related to one specific configuration setting are good candidates.

ULONG id
: the dataspace configuration ID that this gadget will adjust. Use 0 to unregister.

ULONG params
: number of parameters that follow. set this to allow parsing the additional parameters. Older classes had this documented as "flags" and were forced to set it to 0. Currently 3 additional parameters are defined:

CONST_STRPTR title
: the "name" of this gadget. MUI will use that e.g. as title for the popup menu. Can be NULL in which case your popup menu will get a default title. The string is not copied and must remain in memory for the lifetime of your object.

ULONG attr
: the attribute that this gadget adjusts. Can be one of MUI's attributes, such as MUIA_String_Contents or MUIA_Numeric_Value. Can also be one of your own attributes in case you adjust with a "self-made" class. The attribute has to be set()able and get()able as well as support notification.
MUI will use this attribute during its builtin MUIM_Mccprefs_ConfigToGadgets and MUIM_Mccprefs_GadgetsToConfig methods to read/write your supplied configuration ID in its Dataspace.
Also, MUI will setup a notification on attr to allow for realtime preferences updates. Your application (and objects in its tree) will receive a MUIM_UpdateConfig method with the given ID whenever the gadget is changed.
You should always specify the attribute parameter. If you don't, MUI will try to find out the correct attribute automatically by sending OM_GET to your object with several common attributes such as MUIA_Selected (for checkmarks) or MUIA_String_Contents or MUIA_Numeric_Value. As soon as your object's OM_GET method returns TRUE, the tested attribute will be used. This procedure works correctly in many cases and will make older classes support realtime notification, too.

Object *label
: if your gadget has a specific label attached to it (normally a TextObject), pass it here. MUI will try to do fancy things, such as using its MUIA_Text_Contents as popup menu title (when no other title is given). It will also allow the popup menu to work when used on the label itself, not just on the object. Or it could automatically distribute keyboard shortcuts on your prefs page.

## EXAMPLE

```
 obj  DoSuperNew
   fgcolor  PopcolorObject  End
   bgcolor  PopcolorObject  End
   font     PopaslObject  End


obj

 DoMethodobj MUIM_Mccprefs_RegisterGadget
   fgcolor MYCFGID_FGCOLOR

 DoMethodobj MUIM_Mccprefs_RegisterGadget
   ggcolor MYCFGID_BGCOLOR

 DoMethodobj MUIM_Mccprefs_RegisterGadget
   font MYCFGID_FONT
```

## SEE ALSO

MUIM_Mccprefs_GadgetsToConfig, MUIM_Mccprefs_ConfigToGadgets

---