

# MUI 2.3 Autodocs

---

MUI - MagicUserInterface

Version 2.3

(c) Copyright 1993/94 by Stefan Stuntz

- ShareWare -

---

31. Dezember 1994

## **Zusammenfassung**

MUI is an object oriented system to create and maintain graphical user interfaces. From a programmers point of view, using MUI saves a lot of time and makes life much easier. Thinking about complicated terms like window resizing or font sensitivity is simply not necessary.

On the other hand, users of MUI based applications have the ability to customize nearly every pixel of a programs interface according to their personal taste.

MUI ist ein objektorientiertes System zum Erstellen und Verwalten von grafischen Benutzeroberflächen. Vom Standpunkt eines Programmierers aus gesehen spart man mit MUI viel Zeit und Arbeit. Es ist nicht nötig, an sonst so komplizierte Dinge wie Font-Sensitivität oder Window-Resizing auch nur einen Gedanken zu verschwenden.

Auf der anderen Seite hat man als Benutzer einer auf MUI basierenden Applikation die Möglichkeit, nahezu jedes Pixel der Oberfläche an seinen ganz persönlichen Geschmack anzupassen.

L<sup>A</sup>T<sub>E</sub>X-Fassung von: Michael Roth  
Böhringertstr. 13  
78345 Moos

email: mroth@inlet.lake.de  
FIDO: 2:246/8100.14

## Inhaltsverzeichnis

<b>1</b>	<b>muimaster.library</b>	<b>12</b>
1.1	MUI_AllocAslRequest . . . . .	12
1.2	MUI_AslRequest . . . . .	12
1.3	MUI_CreateCustomClass . . . . .	12
1.4	MUI_DeleteCustomClass . . . . .	13
1.5	MUI_DisposeObject . . . . .	14
1.6	MUI_Error . . . . .	14
1.7	MUI_FreeAslRequest . . . . .	15
1.8	MUI_FreeClass . . . . .	15
1.9	MUI_GetClass . . . . .	15
1.10	MUI_MakeObjectA . . . . .	15
1.11	MUI_NewObjectA . . . . .	16
1.12	MUI_Redraw . . . . .	17
1.13	MUI_RequestA . . . . .	19
1.14	MUI_RejectIDCMP . . . . .	20
1.15	MUI_RequestIDCMP . . . . .	20
1.16	MUI_SetError . . . . .	21
<b>2</b>	<b>Application.mui</b>	<b>22</b>
2.1	MUIM_Application_GetMenuCheck . . . . .	22
2.2	MUIM_Application_GetMenuState . . . . .	22
2.3	MUIM_Application_Input . . . . .	22
2.4	MUIM_Application_InputBuffered . . . . .	24
2.5	MUIM_Application_Load . . . . .	25
2.6	MUIM_Application_PushMethod . . . . .	25
2.7	MUIM_Application_ReturnID . . . . .	26
2.8	MUIM_Application_Save . . . . .	27
2.9	MUIM_Application_SetMenuCheck . . . . .	28
2.10	MUIM_Application_SetMenuState . . . . .	28
2.11	MUIM_Application_ShowHelp . . . . .	29
2.12	MUIA_Application_Active . . . . .	29
2.13	MUIA_Application_Author . . . . .	29
2.14	MUIA_Application_Base . . . . .	30
2.15	MUIA_Application_Broker . . . . .	30
2.16	MUIA_Application_BrokerHook . . . . .	30
2.17	MUIA_Application_BrokerPort . . . . .	31
2.18	MUIA_Application_BrokerPri . . . . .	31
2.19	MUIA_Application_Commands . . . . .	31
2.20	MUIA_Application_Copyright . . . . .	32
2.21	MUIA_Application_Description . . . . .	32
2.22	MUIA_Application_DiskObject . . . . .	33
2.23	MUIA_Application_DoubleStart . . . . .	33
2.24	MUIA_Application_DropObject . . . . .	33
2.25	MUIA_Application_ForceQuit . . . . .	34
2.26	MUIA_Application_HelpFile . . . . .	34
2.27	MUIA_Application_Iconified . . . . .	35
2.28	MUIA_Application_Menu . . . . .	36
2.29	MUIA_Application_MenuAction . . . . .	36
2.30	MUIA_Application_MenuHelp . . . . .	36
2.31	MUIA_Application_Menustrip . . . . .	36
2.32	MUIA_Application_RexxHook . . . . .	37
2.33	MUIA_Application_RexxMsg . . . . .	37

2.34	MUIA_Application_RexxString . . . . .	37
2.35	MUIA_Application_SingleTask . . . . .	37
2.36	MUIA_Application_Sleep . . . . .	38
2.37	MUIA_Application_Title . . . . .	38
2.38	MUIA_Application_UseCommodities . . . . .	39
2.39	MUIA_Application_UseRexx . . . . .	39
2.40	MUIA_Application_Version . . . . .	39
2.41	MUIA_Application_Window . . . . .	39
<b>3</b>	<b>Area.mui</b>	<b>40</b>
3.1	MUIM_AskMinMax . . . . .	40
3.2	MUIM_Cleanup . . . . .	40
3.3	MUIM_Draw . . . . .	40
3.4	MUIM_HandleInput . . . . .	41
3.5	MUIM_Hide . . . . .	41
3.6	MUIM_Setup . . . . .	41
3.7	MUIM_Show . . . . .	41
3.8	MUIA_ApplicationObject . . . . .	41
3.9	MUIA_Background . . . . .	42
3.10	MUIA_BottomEdge . . . . .	42
3.11	MUIA_ControlChar . . . . .	43
3.12	MUIA_Disabled . . . . .	43
3.13	MUIA_ExportID . . . . .	43
3.14	MUIA_FixHeight . . . . .	44
3.15	MUIA_FixHeightTxt . . . . .	44
3.16	MUIA_FixWidth . . . . .	44
3.17	MUIA_FixWidthTxt . . . . .	45
3.18	MUIA_Font . . . . .	45
3.19	MUIA_Frame . . . . .	46
3.20	MUIA_FramePhantomHoriz . . . . .	47
3.21	MUIA_FrameTitle . . . . .	47
3.22	MUIA_Height . . . . .	48
3.23	MUIA_HorizWeight . . . . .	48
3.24	MUIA_InnerBottom . . . . .	48
3.25	MUIA_InnerLeft . . . . .	48
3.26	MUIA_InnerRight . . . . .	48
3.27	MUIA_InnerTop . . . . .	49
3.28	MUIA_InputMode . . . . .	49
3.29	MUIA_LeftEdge . . . . .	50
3.30	MUIA_Pressed . . . . .	50
3.31	MUIA_RightEdge . . . . .	50
3.32	MUIA_Selected . . . . .	51
3.33	MUIA_ShowMe . . . . .	51
3.34	MUIA_ShowSelState . . . . .	51
3.35	MUIA_Timer . . . . .	52
3.36	MUIA_TopEdge . . . . .	52
3.37	MUIA_VertWeight . . . . .	52
3.38	MUIA_Weight . . . . .	53
3.39	MUIA_Width . . . . .	53
3.40	MUIA_Window . . . . .	53
3.41	MUIA_WindowObject . . . . .	54

<b>4</b>	<b>Bitmap.mui</b>	<b>54</b>
4.1	MUIA_Bitmap_Bitmap . . . . .	54
4.2	MUIA_Bitmap_Height . . . . .	54
4.3	MUIA_Bitmap_MappingTable . . . . .	55
4.4	MUIA_Bitmap_SourceColors . . . . .	55
4.5	MUIA_Bitmap_Transparent . . . . .	56
4.6	MUIA_Bitmap_Width . . . . .	56
<b>5</b>	<b>Bodychunk.mui</b>	<b>56</b>
5.1	MUIA_Bodychunk_Body . . . . .	56
5.2	MUIA_Bodychunk_Compression . . . . .	57
5.3	MUIA_Bodychunk_Depth . . . . .	57
5.4	MUIA_Bodychunk_Masking . . . . .	57
<b>6</b>	<b>Boopsi.mui</b>	<b>57</b>
6.1	MUIA_Boopsi_Class . . . . .	58
6.2	MUIA_Boopsi_ClassID . . . . .	58
6.3	MUIA_Boopsi_MaxHeight . . . . .	59
6.4	MUIA_Boopsi_MaxWidth . . . . .	59
6.5	MUIA_Boopsi_MinHeight . . . . .	59
6.6	MUIA_Boopsi_MinWidth . . . . .	60
6.7	MUIA_Boopsi_Object . . . . .	60
6.8	MUIA_Boopsi_Remember . . . . .	61
6.9	MUIA_Boopsi_Smart . . . . .	61
6.10	MUIA_Boopsi_TagDrawInfo . . . . .	61
6.11	MUIA_Boopsi_TagScreen . . . . .	62
6.12	MUIA_Boopsi_TagWindow . . . . .	63
<b>7</b>	<b>Coloradjust.mui</b>	<b>63</b>
7.1	MUIA_Coloradjust_Blue . . . . .	63
7.2	MUIA_Coloradjust_Green . . . . .	64
7.3	MUIA_Coloradjust_ModeID . . . . .	64
7.4	MUIA_Coloradjust_Red . . . . .	64
7.5	MUIA_Coloradjust_RGB . . . . .	64
<b>8</b>	<b>Colorfield.mui</b>	<b>65</b>
8.1	MUIA_Colorfield_Blue . . . . .	65
8.2	MUIA_Colorfield_Green . . . . .	65
8.3	MUIA_Colorfield_Pen . . . . .	65
8.4	MUIA_Colorfield_Red . . . . .	66
8.5	MUIA_Colorfield_RGB . . . . .	66
<b>9</b>	<b>Colorpanel.mui</b>	<b>66</b>
<b>10</b>	<b>Cycle.mui</b>	<b>66</b>
10.1	MUIA_Cycle_Active . . . . .	66
10.2	MUIA_Cycle_Entries . . . . .	67
<b>11</b>	<b>Dirlist.mui</b>	<b>67</b>
11.1	MUIM_Dirlist_ReRead . . . . .	68
11.2	MUIA_Dirlist_AcceptPattern . . . . .	68
11.3	MUIA_Dirlist_Directory . . . . .	68
11.4	MUIA_Dirlist_DrawersOnly . . . . .	69
11.5	MUIA_Dirlist_FilesOnly . . . . .	69
11.6	MUIA_Dirlist_FilterDrawers . . . . .	69

11.7	MUIA_Dirlist_FilterHook . . . . .	69
11.8	MUIA_Dirlist_MultiSelDirs . . . . .	70
11.9	MUIA_Dirlist_NumBytes . . . . .	70
11.10	MUIA_Dirlist_NumDrawers . . . . .	70
11.11	MUIA_Dirlist_NumFiles . . . . .	70
11.12	MUIA_Dirlist_Path . . . . .	70
11.13	MUIA_Dirlist_RejectIcons . . . . .	71
11.14	MUIA_Dirlist_RejectPattern . . . . .	71
11.15	MUIA_Dirlist_SortDirs . . . . .	71
11.16	MUIA_Dirlist_SortHighLow . . . . .	71
11.17	MUIA_Dirlist_SortType . . . . .	71
11.18	MUIA_Dirlist_Status . . . . .	72
<b>12</b>	<b>Family.mui</b>	<b>72</b>
12.1	MUIM_Family_AddHead . . . . .	72
12.2	MUIM_Family_AddTail . . . . .	73
12.3	MUIM_Family_Insert . . . . .	73
12.4	MUIM_Family_Remove . . . . .	73
12.5	MUIM_Family_Sort . . . . .	74
12.6	MUIM_Family_Transfer . . . . .	74
12.7	MUIA_Family_Child . . . . .	74
<b>13</b>	<b>Floattext.mui</b>	<b>75</b>
13.1	MUIA_Floattext_Justify . . . . .	75
13.2	MUIA_Floattext_SkipChars . . . . .	75
13.3	MUIA_Floattext_TabSize . . . . .	75
13.4	MUIA_Floattext_Text . . . . .	76
<b>14</b>	<b>Gauge.mui</b>	<b>76</b>
14.1	MUIA_Gauge_Current . . . . .	76
14.2	MUIA_Gauge_Divide . . . . .	77
14.3	MUIA_Gauge_Horiz . . . . .	77
14.4	MUIA_Gauge_InfoText . . . . .	77
14.5	MUIA_Gauge_Max . . . . .	78
<b>15</b>	<b>Group.mui</b>	<b>78</b>
15.1	MUIA_Group_ActivePage . . . . .	78
15.2	MUIA_Group_Child . . . . .	79
15.3	MUIA_Group_Columns . . . . .	79
15.4	MUIA_Group_Horiz . . . . .	80
15.5	MUIA_Group_HorizSpacing . . . . .	80
15.6	MUIA_Group_PageMode . . . . .	81
15.7	MUIA_Group_Rows . . . . .	81
15.8	MUIA_Group_SameHeight . . . . .	82
15.9	MUIA_Group_SameSize . . . . .	82
15.10	MUIA_Group_SameWidth . . . . .	83
15.11	MUIA_Group_Spacing . . . . .	83
15.12	MUIA_Group_VertSpacing . . . . .	83
<b>16</b>	<b>Image.mui</b>	<b>84</b>
16.1	MUIA_Image_FontMatch . . . . .	84
16.2	MUIA_Image_FontMatchHeight . . . . .	84
16.3	MUIA_Image_FontMatchWidth . . . . .	84
16.4	MUIA_Image_FreeHoriz . . . . .	84

16.5	MUIA_Image_FreeVert . . . . .	85
16.6	MUIA_Image_OldImage . . . . .	85
16.7	MUIA_Image_Spec . . . . .	85
16.8	MUIA_Image_State . . . . .	86
<b>17</b>	<b>List.mui</b>	<b>86</b>
17.1	MUIM_List_Clear . . . . .	86
17.2	MUIM_List_Exchange . . . . .	86
17.3	MUIM_List_GetEntry . . . . .	87
17.4	MUIM_List_Insert . . . . .	88
17.5	MUIM_List_InsertSingle . . . . .	88
17.6	MUIM_List_Jump . . . . .	89
17.7	MUIM_List_Move . . . . .	90
17.8	MUIM_List_NextSelected . . . . .	90
17.9	MUIM_List_Redraw . . . . .	91
17.10	MUIM_List_Remove . . . . .	91
17.11	MUIM_List_Select . . . . .	92
17.12	MUIM_List_Sort . . . . .	93
17.13	MUIA_List_Active . . . . .	93
17.14	MUIA_List_AdjustHeight . . . . .	93
17.15	MUIA_List_AdjustWidth . . . . .	93
17.16	MUIA_List_CompareHook . . . . .	94
17.17	MUIA_List_ConstructHook . . . . .	94
17.18	MUIA_List_DestructHook . . . . .	95
17.19	MUIA_List_DisplayHook . . . . .	95
17.20	MUIA_List_Entries . . . . .	96
17.21	MUIA_List_First . . . . .	97
17.22	MUIA_List_Format . . . . .	97
17.23	MUIA_List_InsertPosition . . . . .	99
17.24	MUIA_List_MultiTestHook . . . . .	99
17.25	MUIA_List_Quiet . . . . .	99
17.26	MUIA_List_SourceArray . . . . .	99
17.27	MUIA_List_Title . . . . .	100
17.28	MUIA_List_Visible . . . . .	101
<b>18</b>	<b>Listview.mui</b>	<b>101</b>
18.1	MUIA_Listview_ClickColumn . . . . .	101
18.2	MUIA_Listview_DefClickColumn . . . . .	101
18.3	MUIA_Listview_DoubleClick . . . . .	102
18.4	MUIA_Listview_Input . . . . .	102
18.5	MUIA_Listview_List . . . . .	102
18.6	MUIA_Listview_MultiSelect . . . . .	102
18.7	MUIA_Listview_ScrollerPos . . . . .	103
18.8	MUIA_Listview_SelectChange . . . . .	103
<b>19</b>	<b>Menustrip.mui</b>	<b>103</b>
19.1	MUIA_Menustrip_Enabled . . . . .	103
<b>20</b>	<b>Menu.mui</b>	<b>104</b>
20.1	MUIA_Menu_Enabled . . . . .	104
20.2	MUIA_Menu_Title . . . . .	104

<b>21MenuItem.mui</b>	<b>104</b>
21.1 MUIA_Menuitem_Checked . . . . .	104
21.2 MUIA_Menuitem_Checkit . . . . .	104
21.3 MUIA_Menuitem_Enabled . . . . .	105
21.4 MUIA_Menuitem_Exclude . . . . .	105
21.5 MUIA_Menuitem_Shortcut . . . . .	105
21.6 MUIA_Menuitem_Title . . . . .	105
21.7 MUIA_Menuitem_Toggle . . . . .	105
21.8 MUIA_Menuitem_Trigger . . . . .	106
<b>22Notify.mui</b>	<b>106</b>
22.1 MUIM_CallHook . . . . .	106
22.2 MUIM_FindUData . . . . .	107
22.3 MUIM_GetUData . . . . .	107
22.4 MUIM_KillNotify . . . . .	108
22.5 MUIM_MultiSet . . . . .	108
22.6 MUIM_NoNotifySet . . . . .	109
22.7 MUIM_Notify . . . . .	109
22.8 MUIM_Set . . . . .	111
22.9 MUIM_SetAsString . . . . .	112
22.10MUIM_SetUData . . . . .	112
22.11MUIM_WriteLong . . . . .	113
22.12MUIM_WriteString . . . . .	114
22.13MUIA_AppMessage . . . . .	114
22.14MUIA_HelpFile . . . . .	115
22.15MUIA_HelpLine . . . . .	115
22.16MUIA_HelpNode . . . . .	115
22.17MUIA_NoNotify . . . . .	116
22.18MUIA_Revision . . . . .	116
22.19MUIA_UserData . . . . .	116
22.20MUIA_Version . . . . .	116
<b>23Palette.mui</b>	<b>117</b>
23.1 MUIA_Palette_Entries . . . . .	117
23.2 MUIA_Palette_Groupable . . . . .	118
23.3 MUIA_Palette_Names . . . . .	118
<b>24Popasl.mui</b>	<b>119</b>
24.1 MUIA_Popasl_Active . . . . .	119
24.2 MUIA_Popasl_StartHook . . . . .	120
24.3 MUIA_Popasl_StopHook . . . . .	121
24.4 MUIA_Popasl_Type . . . . .	121
<b>25Poplist.mui</b>	<b>121</b>
25.1 MUIA_Poplist_Array . . . . .	121
<b>26Popobject.mui</b>	<b>121</b>
26.1 MUIA_Popobject_Follow . . . . .	122
26.2 MUIA_Popobject_Light . . . . .	122
26.3 MUIA_Popobject_Object . . . . .	122
26.4 MUIA_Popobject_ObjStrHook . . . . .	122
26.5 MUIA_Popobject_StrObjHook . . . . .	123
26.6 MUIA_Popobject_Volatile . . . . .	124
26.7 MUIA_Popobject_WindowHook . . . . .	124



<b>27Popstring.mui</b>	<b>124</b>
27.1 MUIM_Popstring_Close . . . . .	125
27.2 MUIM_Popstring_Open . . . . .	125
27.3 MUIA_Popstring_Button . . . . .	125
27.4 MUIA_Popstring_CloseHook . . . . .	126
27.5 MUIA_Popstring_OpenHook . . . . .	126
27.6 MUIA_Popstring_String . . . . .	127
27.7 MUIA_Popstring_Toggle . . . . .	127
<b>28Prop.mui</b>	<b>127</b>
28.1 MUIA_Prop_Entries . . . . .	127
28.2 MUIA_Prop_First . . . . .	128
28.3 MUIA_Prop_Horiz . . . . .	128
28.4 MUIA_Prop_Slider . . . . .	128
28.5 MUIA_Prop_Visible . . . . .	128
<b>29Radio.mui</b>	<b>128</b>
29.1 MUIA_Radio_Active . . . . .	128
29.2 MUIA_Radio_Entries . . . . .	129
<b>30Rectangle.mui</b>	<b>129</b>
30.1 MUIA_Rectangle_HBar . . . . .	129
30.2 MUIA_Rectangle_VBar . . . . .	130
<b>31Register.mui</b>	<b>130</b>
31.1 MUIA_Register_Frame . . . . .	130
31.2 MUIA_Register_Titles . . . . .	130
<b>32Scale.mui</b>	<b>131</b>
32.1 MUIA_Scale_Horiz . . . . .	131
<b>33Scrmodelist.mui</b>	<b>131</b>
<b>34Scrollbar.mui</b>	<b>131</b>
<b>35Scrollgroup.mui</b>	<b>131</b>
35.1 MUIA_Scrollgroup_Contents . . . . .	132
35.2 MUIA_Scrollgroup_FreeHoriz . . . . .	132
35.3 MUIA_Scrollgroup_FreeVert . . . . .	132
<b>36Slider.mui</b>	<b>132</b>
36.1 MUIA_Slider_Level . . . . .	132
36.2 MUIA_Slider_Max . . . . .	132
36.3 MUIA_Slider_Min . . . . .	133
36.4 MUIA_Slider_Quiet . . . . .	133
36.5 MUIA_Slider_Reverse . . . . .	133
<b>37String.mui</b>	<b>133</b>
37.1 MUIA_String_Accept . . . . .	133
37.2 MUIA_String_Acknowledge . . . . .	134
37.3 MUIA_String_AttachedList . . . . .	134
37.4 MUIA_String_BufferPos . . . . .	134
37.5 MUIA_String_Contents . . . . .	134
37.6 MUIA_String_DisplayPos . . . . .	135
37.7 MUIA_String_EditHook . . . . .	135

37.8	MUIA_String_Format . . . . .	135
37.9	MUIA_String_Integer . . . . .	136
37.10	MUIA_String_MaxLen . . . . .	136
37.11	MUIA_String_Reject . . . . .	136
37.12	MUIA_String_Secret . . . . .	136
<b>38</b>	<b>Text.mui</b>	<b>137</b>
38.1	MUIA_Text_Contents . . . . .	137
38.2	MUIA_Text_HiChar . . . . .	138
38.3	MUIA_Text_PreParse . . . . .	138
38.4	MUIA_Text_SetMax . . . . .	138
38.5	MUIA_Text_SetMin . . . . .	139
<b>39</b>	<b>Virtgroup.mui</b>	<b>139</b>
39.1	MUIA_Virtgroup_Height . . . . .	139
39.2	MUIA_Virtgroup_Left . . . . .	139
39.3	MUIA_Virtgroup_Top . . . . .	140
39.4	MUIA_Virtgroup_Width . . . . .	140
<b>40</b>	<b>Volumelist.mui</b>	<b>140</b>
<b>41</b>	<b>Window.mui</b>	<b>140</b>
41.1	MUIM_Window_GetMenuCheck . . . . .	141
41.2	MUIM_Window_GetMenuState . . . . .	141
41.3	MUIM_Window_ScreenToBack . . . . .	141
41.4	MUIM_Window_ScreenToFront . . . . .	141
41.5	MUIM_Window_SetCycleChain . . . . .	142
41.6	MUIM_Window_SetMenuCheck . . . . .	142
41.7	MUIM_Window_SetMenuState . . . . .	143
41.8	MUIM_Window_ToBack . . . . .	143
41.9	MUIM_Window_ToFront . . . . .	143
41.10	MUIA_Window_Activate . . . . .	144
41.11	MUIA_Window_ActiveObject . . . . .	144
41.12	MUIA_Window_AltHeight . . . . .	144
41.13	MUIA_Window_AltLeftEdge . . . . .	144
41.14	MUIA_Window_AltTopEdge . . . . .	145
41.15	MUIA_Window_AltWidth . . . . .	145
41.16	MUIA_Window_AppWindow . . . . .	145
41.17	MUIA_Window_Backdrop . . . . .	146
41.18	MUIA_Window_Borderless . . . . .	146
41.19	MUIA_Window_CloseGadget . . . . .	146
41.20	MUIA_Window_CloseRequest . . . . .	146
41.21	MUIA_Window_DefaultObject . . . . .	146
41.22	MUIA_Window_DepthGadget . . . . .	147
41.23	MUIA_Window_DragBar . . . . .	147
41.24	MUIA_Window_FancyDrawing . . . . .	147
41.25	MUIA_Window_Height . . . . .	147
41.26	MUIA_Window_ID . . . . .	148
41.27	MUIA_Window_InputEvent . . . . .	148
41.28	MUIA_Window_LeftEdge . . . . .	149
41.29	MUIA_Window_Menu . . . . .	149
41.30	MUIA_Window_MenuAction . . . . .	149
41.31	MUIA_Window_Menustrip . . . . .	150
41.32	MUIA_Window_MouseObject . . . . .	150

41.33MUIA_Window_NeedsMouseObject . . . . .	150
41.34MUIA_Window_NoMenus . . . . .	150
41.35MUIA_Window_Open . . . . .	151
41.36MUIA_Window_PublicScreen . . . . .	151
41.37MUIA_Window_RefWindow . . . . .	151
41.38MUIA_Window_RootObject . . . . .	152
41.39MUIA_Window_Screen . . . . .	152
41.40MUIA_Window_ScreenTitle . . . . .	152
41.41MUIA_Window_SizeGadget . . . . .	153
41.42MUIA_Window_SizeRight . . . . .	153
41.43MUIA_Window_Sleep . . . . .	153
41.44MUIA_Window_Title . . . . .	153
41.45MUIA_Window_TopEdge . . . . .	153
41.46MUIA_Window_Width . . . . .	154
41.47MUIA_Window_Window . . . . .	154

## 1 muimaster.library

### PURPOSE

muimaster.library contains functions for creating and disposing objects, for requester handling and for controlling custom classes. Additionally, several of the standard MUI classes are built into muimaster.library. For you as a programmer, there is no difference between using a builtin class or an external class coming as "sys:classes/!foobar!.mui". The MUI object generation call takes care of this situation and loads external classes automatically when they are needed.

### 1.1 MUI\_AllocAslRequest

#### FUNCTION

Provide an interface to asl.library. Using this ensures your application will benefit from future expansions to MUI's window and iconification handling.

#### SEE ALSO

asl.library/AllocAslRequest

### 1.2 MUI\_AslRequest

#### FUNCTION

Provide an interface to asl.library. Using this ensures your application will benefit from future expansions to MUI's window and iconification handling.

#### SEE ALSO

asl.library/AslRequest

### 1.3 MUI\_CreateCustomClass – create a public/private custom class.

#### SYNOPSIS

MUI\_CreateCustomClass (base, supername, supermcc, datasize, dispfunc)  
                           A0   A1           A2           D0       A3

```
struct MUI_CustomClass * MUI_CreateCustomClass( struct Library *, char *,
int, APTR );
```

#### FUNCTION

This function creates a public or private MUI custom class. Public custom classes are shared libraries and can be found in "libs:mui/!foobar!.mcc". Private classes simply consist of a dispatcher and are built into applications.

MUI\_CreateCustomClass() returns a pointer to a struct MUI\_CustomClass which in turn contains a pointer to a struct IClass. For private classes, this struct IClass pointer needs to be fed to a intuition.library/NewObject() call to create new objects.

MUI creates the dispatcher hook for you, you may **not** use the IClass-!cl\_Dispatcher.h\_Data field! If you need custom data for your dispatcher, use the

`cl_UserData` of the `IClass` structure or the `mcc_UserData` of the `MUI_CustomClass` structure.

For public classes, MUI makes sure that `a6` contains a pointer to your library base when your dispatcher is called. For private classes, you will need to keep track of `A4` of similar things your compiler may need yourself.

#### INPUTS

**base** = if you create a public class, you have to call `MUI_CreateCustomClass()` from your libraries init function. In this case, place your library base pointer here. For private classes, you must supply `NULL`.

**supername** = super class of your class. This can either be a builtin MUI class ("`xyz.mui`") or a external custom class ("`xyz.mcc`").

**supermcc** = if (and only if) the super class is a private custom class and hence has no name, you are allowed to pass a `NULL` supername and a pointer to the `MUI_CustomClass` structure of the super class here.

**datasize** = size of your classes data structure.

**dispfunc** = your classes dispatcher function (no hook!). The dispatcher will be called with a struct `IClass` in `a0`, with your object in `a2` and the message in `a1`.

#### RESULT

A pointer to a struct `MUI_CustomClass` or `NULL` to indicate an error.

#### SEE ALSO

`MUI_DeleteCustomClass()`

### 1.4 *MUI\_DeleteCustomClass* – delete a public/private custom class.

#### SYNOPSIS

```
MUI_DeleteCustomClass ( mcc )
                      A0
```

```
BOOL MUI_DeleteCustomClass(struct MUI_CustomClass *);
```

#### FUNCTION

Delete a public or private custom class. Note that you must not delete classes with outstanding objects or sub classes.

#### INPUTS

**mcc** = pointer obtained from `MUI_CreateCustomClass()`.

#### RESULT

`TRUE` if all went well, `FALSE` if some objects or sub classes were still hanging around. Nothing will be freed in this case.

**SEE ALSO**

MUI\_CreateCustomClass()

**1.5 MUI\_DisposeObject – Delete a MUI object.****SYNOPSIS**

```
MUI_DisposeObject( object )
                  A0

VOID MUI_DisposeObject( APTR );
```

**FUNCTION**

Deletes a MUI object and all of its auxiliary data. These objects are all created by MUI\_NewObject(). Objects of certain classes "own" other objects, which will also be deleted when the object is passed to MUI\_DisposeObject(). Read the per-class documentation carefully to be aware of these instances.

**INPUTS**

**object** = abstract pointer to a MUI object returned by MUI\_NewObject(). The pointer may be NULL, in which case this function has no effect.

**RESULT**

None.

**SEE ALSO**

MUI\_NewObject(), SetAttrs(), GetAttr().

**1.6 MUI\_Error – Return extra information from the MUI system.****SYNOPSIS**

```
LONG MUI_Error(VOID);
```

**FUNCTION**

Some MUI functions will set an error if they fail for some reason. The error function is task sensitive, only the task that caused the error will receive it from this function.

**RESULT**

Currently, the following error values are defined:

MUIE_OK	- no error, everything alright.
MUIE_OutOfMemory	- went out of memory.
MUIE_OutOfGfxMemory	- went out of graphics memory.
MUIE_InvalidWindowObject	- NULL window specified.
MUIE_MissingLibrary	- can't open a needed library.
MUIE_NoARexx	- unable to create arexx port.
MUIE_SingleTask	- application is already running.

**SEE ALSO**

`MUI_SetError()`

**1.7 MUIFreeAslRequest****FUNCTION**

Provide an interface to `asl.library`. Using this ensures your application will benefit from future expansions to MUI's window and iconification handling.

**SEE ALSO**

`asl.library/FreeAslRequest`

**1.8 MUIFreeClass – Free class.****SYNOPSIS**

```
MUIFreeClass( classptr )
               A0
```

```
VOID MUIFreeClass(struct IClass *classptr);
```

**FUNCTION**

This function is obsolete since MUI V8. Use `MUI_DeleteCustomClass()` instead.

**SEE ALSO**

`MUI_CreateCustomClass()`, `MUI_DeleteCustomClass()`

**1.9 MUIGetClass – Get a pointer to a MUI class.****SYNOPSIS**

```
class = MUIGetClass( classid )
D0                      A0
```

```
struct IClass * MUIGetClass(char *classid);
```

**FUNCTION**

This function is obsolete since MUI V8. Use `MUI_CreateCustomClass` instead.

**SEE ALSO**

`MUI_CreateCustomClass()`, `MUI_DeleteCustomClass()`

**1.10 MUIMakeObjectA – create an object from the builtin object collection.**

`MUIMakeObject` – Varargs stub for `MUIMakeObjectA`

**SYNOPSIS**

```
object = MUI_MakeObjectA( objtype, params )
D0                                D0      A0
```

```
Object * MUI_MakeObjectA(ULONG type, ULONG *params);
Object * MUI_MakeObject(ULONG type, ...);
```

**FUNCTION**

Prior to muimaster.library V8, MUI was distributed with several macros to help creating often used objects. This practice was easy, but using lots of these macros often resulted in big programs. Now, muimaster library contains an object library with several often used objects already built in.

MUI\_MakeObject() takes the type of the object as first parameter and a list of additional (type specific) parameters. Note that these additional values are **not** a taglist!

See the header file mui.h for documentation on object types and the required parameters.

**SEE ALSO**

MUI\_CreateCustomClass(), MUI\_DeleteCustomClass()

**1.11 MUI\_NewObjectA – Create an object from a class.**

MUI\_NewObject – Varargs stub for MUI\_NewObjectA().

**SYNOPSIS**

```
object = MUI_NewObjectA( class, tags )
D0                                A0    A1
```

```
APTR MUI_NewObjectA( char *, struct TagItem * );
object = MUI_NewObject( classID, Tag1, ... )
APTR MUI_NewObject( classID, ULONG, ... );
```

**FUNCTION**

This is the general method of creating objects from MUI classes. You specify a class by its ID string. If the class is not already in memory or built into muimaster.library, it will be loaded using `OpenLibrary("mui/%s",0)`.

You further specify initial "create-time" attributes for the object via a TagItem list, and they are applied to the resulting generic data object that is returned. The attributes, their meanings, attributes applied only at create-time, and required attributes are all defined and documented on a class-by-class basis.

**INPUTS**

**classID** = the name/ID string of a MUI class, e.g. "Image.mui". Class names are case sensitive!

**tagList** = pointer to array of TagItems containing attribute/value pairs to be applied to the object being created.



**RESULT**

A MUI object, which may be used in different contexts such as an application, window or gadget, and may be manipulated by generic functions. You eventually free the object using `MUI_DisposeObject()`. `NULL` indicates failure, more information on the error can be obtained with `MUI_Error()`.

**SEE ALSO**

`MUI_DisposeObject()`, `MUI_Error()`, `SetAttrs()`, `GetAttr()`.

**1.12 MUI\_Redraw – Redraw yourself.****SYNOPSIS**

```
MUI_Redraw( obj, flag )
           A0 D0
```

```
VOID MUI_Redraw( Object *obj, ULONG flag );
```

**FUNCTION**

With `MUI_Redraw()`, an object tells itself to refresh, e.g. when some internal attributes were changed. Calling `MUI_Redraw()` is only legal within a custom class dispatcher, using this function within an applications main part is invalid!

Most objects graphical representation in a window depends on some attributes. A fuel gauge for example would depend on its `MUIA_Gauge_Current` attribute, an animation object would depend on `MUIA_Animation_CurrentFrame`.

Whenever someone changes such an attribute with a `SetAttrs()` call, the corresponding object receives an `OM_SET` method with the new value. Usually, it could just render itself with some `graphics.library` calls. However, if the object is placed in a virtual group or if some other clipping or coordinate translation is required, this simple rendering will lead into problems.

That's why MUI offers the `MUI_Redraw()` function call. Instead of drawing directly during `OM_SET`, you should simply call `MUI_Redraw()`. MUI calculates all necessary coordinates and clip regions (in case of virtual groups) and then sends a `MUIM_Draw` method to your object.

To emphasize this point again: The only time your object is allowed to render something is when you receive a `MUIM_Draw` method. Drawing during other methods is illegal.

**NOTE**

As long as no special cases (e.g. virtual groups) are present, `MUI_Redraw` is very quick and calls your `MUIM_Draw` method immediately. No coordinate translations or clip regions need to be calculated.

**INPUTS**

**obj** - pointer to yourself.

**flag** - `MADF_DRAWOBJECT` or `MADF_DRAWUPDATE`. The flag given here affects the objects flags when MUI calls the `MUIM_Draw` method. There are several caveats when implementing `MUIM_DRAW`, see the developer documentation for details.

**EXAMPLE**

```

/* Note: This example was broken up to version 2.1 of muimaster.doc */

LONG mSet(struct IClass *cl, Object *obj, struct opSet *msg)
{
    struct Data *data = INST_DATA(cl, obj);
    struct TagItem *tags, *tag;

    for (tags=msg->ops_AttrList; tag=NextTagItem(&tags);)
    {
        switch (tag->ti_Tag)
        {
            case MYATTR_PEN_1:
                data->pen1 = tag->ti_Data;      /* set the new value */
                data->mark = 1;                 /* set internal marker*/
                MUI_Redraw(obj, MADF_DRAWUPDATE); /* update ourselves */
                break;

            case MYATTR_PEN_2:
                data->pen2 = tag->ti_Data;      /* set the new value */
                data->mark = 2;                 /* set internal marker*/
                MUI_Redraw(obj, MADF_DRAWUPDATE); /* update ourselves */
                break;
        }
    }

    return(DoSuperMethodA(cl, obj, msg));
}

LONG mDraw(struct IClass *cl, Object *obj, struct MUIP_Draw *msg)
{
    struct Data *data = INST_DATA(cl, obj);

    // ** Note: You *must* call the super method prior to do
    // ** anything else, otherwise msg->flags will not be set
    // ** properly !!!

    DoSuperMethodA(cl, obj, msg); // ALWAYS REQUIRED!

    if (msg->flags & MADF_DRAWUPDATE)
    {
        /* called as a result of our MUI_Redraw() during
           MUIM_Set method. Depending on our internal
           marker, we render different things. */

        switch (data->mark)
        {
            case 1: RenderChangesFromPen1(cl, obj); break;
            case 2: RenderChangesFromPen2(cl, obj); break;
        }
    }
    else if (msg->flags & MADF_DRAWOBJECT)
    {
        /* complete redraw, maybe the window was just opened. */
        DrawObjectCompletely(cl, obj);
    }
}

```

```

/* if MADF_DRAWOBJECT wasn't set, MUI just wanted to update
   the frame or some other part of our object. In this case
   we just do nothing. */

return(0);
}

```

**SEE ALSO**

area.mui/MUIM\_Draw

**1.13 MUI\_RequestA – Pop up a MUI requester.****SYNOPSIS**

```

MUI_RequestA(app,win,flags,title,gadgets,format,params)
               D0 D1 D2  A0 A1      A2   A3

```

```

LONG MUI_RequestA ( APTR app, APTR win, LONGBITS flags, char *title,
char *gadgets, char *format, APTR params );

```

```

LONG MUI_Request ( APTR app, APTR win, LONGBITS flags, char *title,
char *gadgets, char *format, ...);

```

**FUNCTION**

Pop up a MUI requester. Using a MUI requester instead of a standard system requester offers you the possibility to include text containing all the text engine format codes.

**INPUTS**

**app** - The application object. If you leave this NULL, MUI\_RequestA() will fall back to a standard system requester.

**win** - Pointer to a window of the application. If this is used, the requester will appear centered relative to this window.

**flags** - For future expansion, must be 0 for now.

**title** - Title for the requester window. Defaults to the name of the application when NULL (and app!=NULL).

**gadgets** - Pointer to a string containing the possible answers. The format looks like "Save—Use—Test—Cancel". If you precede an entry with a '\*', this answer will become the active object. Pressing `⏎` will terminate the requester with this response. A '\_' character indicates the keyboard shortcut for this response.

**format** - A printf-style formatting string.

**params** - Pointer to an array of ULONG containing the parameter values for format.

**RESULT**

0, 1, ..., N = Successive id values, for the gadgets you specify for the requester.

**NOTE:** The numbering from left to right is actually: 1, 2, ..., N, 0. In case of a problem (e.g. out of memory), the function returns FALSE.

**SEE ALSO**

MUIA\_Text\_Contents

### 1.14 MUI\_RejectIDCMP – Reject previously requested input events.

**SYNOPSIS**

```
MUI_RejectIDCMP( obj, flags )
                A0 D0
```

```
VOID MUI_RejectIDCMP( Object *obj, ULONG flags );
```

**FUNCTION**

Reject previously requested input events. You should ensure that you reject all input events you requested for an object before it gets disposed. Rejecting flags that you never requested has no effect.

Critical flags such as IDCMP\_MOUSEMOVE and IDCMP\_INTUITICKS should be rejected as soon as possible. See MUI\_RequestIDCMP() for details.

**INPUTS**

**obj** - pointer to yourself as an object.

**flags** - one or more IDCMP\_XXXX flags.

**EXAMPLE**

```
LONG CleanupMethod(struct IClass *cl, Object *obj, Msg msg)
{
    MUI_RejectIDCMP( obj, IDCMP_MOUSEBUTTONS|IDCMP_RAWKEY );
    return(DoSuperMethodA(cl,obj,msg));
}
```

**SEE ALSO**

MUI\_RequestIDCMP()

### 1.15 MUI\_RequestIDCMP – Request input events for your custom class.

**SYNOPSIS**

```
MUI_RequestIDCMP( obj, flags )
                A0 D0
```

```
VOID MUI_RequestIDCMP( Object *obj, ULONG flags );
```

**FUNCTION**

If your custom class needs to do some input handling, you must explicitly request the events you want to receive. You can request (and reject) events at any time.

Whenever an input event you requested arrives at your parent windows message port, your object will receive a MUI\_HandleInput method.

**NOTE**

Time consuming IDCMP flags such as IDCMP\_INTUITICKS and IDCMP\_MOUSEMOVE should be handled with care. Too many objects receiving them will degrade performance. With the following paragraph in mind, this isn't really a problem:

You should try to request critical events only when you really need them and reject them with MUI\_RejectIDCMP() as soon as possible. Usually, mouse controlled objects only need MOUSEMOVES and INTUITICKS when a button is pressed. You should request these flags only on demand, i.e. after receiving a mouse down event and reject them immediately after the button has been released.

**INPUTS**

**obj** - pointer to yourself as an object.

**flags** - one or more IDCMP\_XXXX flags.

**EXAMPLE**

```
LONG SetupMethod(struct IClass *cl, Object *obj, Msg msg)
{
    if (!DoSuperMethodA(cl,obj,msg))
        return(FALSE);

    /* do some setup here... */
    ...;

    /* i need mousebutton events and keyboard */
    MUI_RequestIDCMP( obj, IDCMP_MOUSEBUTTONS|IDCMP_RAWKEY );

    return(TRUE);
}
```

**SEE ALSO**

MUI\_RejectIDCMP()

**1.16 MUI\_SetError – Set an error value.****SYNOPSIS**

```
VOID MUI_SetError(LONG);
```

**FUNCTION**

Setup a MUI error. MUI\_Error() will return this value when asked.

**SEE ALSO**

MUI\_Error()

## 2 Application.mui

Application class is the master class for all MUI applications. It serves as a kind of anchor for all input, either coming from the user or somewhere from the system, e.g. commodities or ARexx messages.

An application can have any number of sub windows, these windows are the children of the application.

### 2.1 MUIM\_Application\_GetMenuCheck (V4) (OBSOLETE)

**SYNOPSIS**

DoMethod(obj,MUIM\_Application\_GetMenuCheck,ULONG MenuID);

**FUNCTION**

Ask whether a checkmark menu item has its checkmark set or cleared. The application will ask its sub windows for a menu item with the given id and return the state of the first item it finds.

**INPUTS**

**MenuID** - the value you wrote into the UserData field of struct NewMenu.

**SEE ALSO**

MUIM\_Application\_SetMenuCheck, MUIA\_Application\_Menu

### 2.2 MUIM\_Application\_GetMenuState (V4) (OBSOLETE)

**SYNOPSIS**

DoMethod(obj,MUIM\_Application\_GetMenuState,ULONG MenuID);

**FUNCTION**

Ask whether a menu item is enabled or disabled. The application will ask its sub windows for a menu item with the given id and return the state of the first item it finds.

**INPUTS**

**MenuID** - the value you wrote into the UserData field of struct NewMenu.

**SEE ALSO**

MUIM\_Application\_SetMenuState, MUIA\_Application\_Menu

### 2.3 MUIM\_Application\_Input (V4)

**SYNOPSIS**

DoMethod(obj,MUIM\_Application\_Input,LONGBITS \*signal);

## FUNCTION

The MUI system itself does not wait for any user input. It just tells your application which signal bits it has allocated, then it's up to you to call MUI's input handle function when one of these signals gets set.

In a simple MUI application you would just `Wait()` for these signals and call MUI when one is received. However, you can perfectly allocate some signal bits yourself and include them in your `Wait()` command. You needn't even `Wait()`, your application could maybe calculate some fractal graphics or copy disks, the only important thing is that you call MUI's input method when one of the MUI allocated signals arrives.

The usual way of communication with your user interface is via return ids. Every action happening to the GUI can create return ids, e.g. pressing a button or trying to close a window. MUI buffers these ids and uses them as result codes for the input method. That's where you can get it from and take the appropriate actions.

Now let's have a look on a usual input loop of a MUI application. Imagine you have an `Play` and a `Cancel` button and have previously told them to return `ID_PLAY` and `ID_CANCEL` when pressed. (see `MUIM_Notify` and `MUIM_Application_ReturnID` on information about these topics). Your input loop would look like this:

```
while (running)
{
    ULONG signals;

    switch (DoMethod(app,MUIM_Application_Input,&signals))
    {
        case ID_PLAY:
            PlaySound();
            break;

        case ID_CANCEL:
        case MUIV_Application_ReturnID_Quit:
            running = FALSE;
            break;
    }

    if (running && signals) Wait(signals);
}
```

So what is happening here?

First, you have to call the `MUIM_Application_Input` method. You supply the address of a `ULONG` as parameter, that's where MUI fills in the signals it needs. Note that you can call the input method at any time, regardless of signal setting. MUI will simply return when there is nothing to do.

In case the user pressed the `Play` or the `Cancel` button, `MUIM_Application_Input` will return `ID_PLAY` or `ID_CANCEL`. Otherwise you will receive a 0, that's why you cannot use 0 as one of your id values.

There is one predefined id called `MUIV_Application_ReturnID_Quit`. This will be sent to you when someone tried to quit your application from outside, e.g. via commodities exchange or the `ARexx "quit"` command. It is required that your application handles this id, just treat as if the user clicked on a "Quit" button or selected a "Quit" menu item.

After handling the return value, you have to examine if MUI wants you to wait for any signals. If this is the case (`signals != 0`), just wait for it. If MUI puts a 0

into signals it wants to tell you to immediately call the input method again, maybe some other return ids have received and need to be handled. You *must* check this because Wait()ing on a zero signal mask is not a good idea!

#### NOTE

It is very important that you call the input method whenever a signal arrives. MUI needs this to correctly refresh its windows, handle resizing and iconification operations and commodities and ARexx messages. If you don't, you will annoy your user!

If your program needs to be in a state where you are for some reasons unable to call the input method for a considerable amount of time (maybe half a second or more), you should put your application to sleep. See MUIA\_Application\_Sleep on how to do this.

#### SEE ALSO

MUIA\_Application\_Sleep, MUIM\_Application\_InputBuffered

### 2.4 MUIM\_Application\_InputBuffered (V4)

#### SYNOPSIS

```
DoMethod(obj,MUIM_Application_InputBuffered,);
```

#### FUNCTION

Imagine your application does some time consuming operation, e.g. copying a disk, and you are for some reasons unable to react on return ids during this period. One solution would be to simply put your application to sleep, it will get a busy pointer and the user knows whats going on.

However, this will make it impossible for the user to resize your applications windows or iconify it, he will have to wait until you are done with your operation.

MUIM\_Application\_InputBuffered offers a solution for this problem. Using this method, you needn't set to sleep your application. Just call it on a regular basis and MUI will be able to handle all actions concerning the GUI. You do not need to pay attention on return values, they remain on an internal stack until your next call to the non buffered input method.

#### EXAMPLE

```
for (track=0; track<80; track++)
{
    read_track();
    DoMethod(app,MUIM_Application_InputBuffered);
    write_track();
    DoMethod(app,MUIM_Application_InputBuffered);
}
```

#### SEE ALSO

MUIM\_Application\_Input, MUIA\_Application\_Sleep



## 2.5 MUIM\_Application\_Load (V4)

### SYNOPSIS

DoMethod(obj,MUIM\_Application\_Load,STRPTR name);

### FUNCTION

MUIM\_Application\_Save, MUIM\_Application\_Load and MUIA\_ExportID offer an easy way of saving and loading a programs configuration.

Each gadget with a non NULL MUIA\_ExportID will get its contents saved during MUIM\_Application\_Save and restored during MUIM\_Application\_Load. This makes it very easy to design a configuration window with "Save", "Use" and "Cancel" buttons to allow the user storing the settings. When the application starts, you would just have to call MUIM\_Application\_Load and the stored settings will be read and installed.

Not all classes are able to import and export their contents. Currently, you may define MUIA\_ExportIDs for

String class	-	MUIA_String_Contents is ex/imported.
Radio class	-	MUIA_Radio_Active is ex/imported.
Cycle class	-	MUIA_Cycle_Active is ex/imported.
List class	-	MUIA_List_Active is /ex/imported.
Text class	-	MUIA_Text_Contents is ex/imported.
Slider class	-	MUIA_Slider_Level is ex/imported.
Area class	-	MUIA_Selected is ex/imported (e.g. for Checkmark gadgets)
Menuitem class	-	MUIA_Checked is ex/imported (V9).
Group class	-	MUIA_Group_ActivePage is ex/imported (V8).

### INPUTS

**name** - Name of the file you wish to load the settings from. Usually you won't need to think of a real name but instead use one of the magic cookies

- MUIV\_Application\_Load\_ENV or
- MUIV\_Application\_Load\_ENVARC.

### EXAMPLE

see the sample program "Settings.c"

### SEE ALSO

MUIM\_Application\_Save, MUIA\_ExportID

## 2.6 MUIM\_Application\_PushMethod (V4)

### SYNOPSIS

DoMethod(obj,MUIM\_Application\_PushMethod,Object \*dest, LONG count, /\* ... \*/);

**FUNCTION**

Usually, you may not talk to the MUI system from two tasks at the same time. MUIM\_Application\_PushMethod provides some kind of solution for this problem.

This (and only this) method may be called from a second task. It takes another method as parameter and puts in onto a private stack of the application object. The next time MUIM\_Application\_Input is called, the pushed method will be executed in the context of the current task.

**INPUTS**

**dest** - object on which to perform the pushed method.

**count** - number of following arguments.

... - the destination method.

**EXAMPLE**

```
/* set a status line from a sub task */
DoMethod(app,MUIM_Application_PushMethod,
    txstatus,3,MUIM_Set,MUIA_Text_Contents,"reading...");
```

**SEE ALSO**

MUIM\_Application\_Input

**2.7 MUIM\_Application\_ReturnID (V4)****SYNOPSIS**

```
DoMethod(obj,MUIM_Application_ReturnID,ULONG retid);
```

**FUNCTION**

Tell MUI to return the given id with the next call to MUIM\_Application\_Input.

Together with the MUI's notification mechanism, this method connects your user interface and your program. If you e.g. want to be informed if the user presses a "Play" button, you would have define an id for this action and set up a notification event with MUIM\_Notify.

You can use any long word as return id, except from -255 up to 0. These values are reserved for MUI's internal use and for special return values like MUIM\_Application\_ReturnID\_Quit.

Note that MUI will put all incoming return ids onto a private fifo stack and feed this stack to its input methods result code later.

**EXAMPLE**

```
/* inform me if a button is pressed (actually released, */
/* since this is the way amiga buttons are handled)      */
#define ID_PLAYBUTTON 42

...

DoMethod(buttonobj, MUIM_Notify,
```

```

    MUIA_Pressed, FALSE,
    appobj, 2, MUIM_Application_ReturnID, ID_PLAYBUTTON);

...

while (running)
{
    switch (DoMethod(appobj,MUIM_Application_Input,&sigs))
    {
        case ID_PLAYBUTTON:
            printf("Ok, lets play a game...");
            break;
    }
}

```

**SEE ALSO**

MUIM\_Application\_Input, MUIM\_Notify

**2.8 MUIM\_Application\_Save (V4)****SYNOPSIS**

DoMethod(obj,MUIM\_Application\_Save,STRPTR name);

**FUNCTION**

MUIM\_Application\_Save, MUIM\_Application\_Load and MUIA\_ExportID offer an easy way of saving and loading a programs configuration.

Each gadget with a non NULL MUIA\_ExportID will get its contents saved during MUIM\_Application\_Save and restored during MUIM\_Application\_Load. This makes it very easy to design a configuration window with "Save", "Use" and "Cancel" buttons to allow the user storing the settings. When the application starts, you would just have to call MUIM\_Application\_Load and the stored settings will be read and installed.

Not all classes are able to import and export their contents. Currently, you may define MUIA\_ExportIDs for

String class	- MUIA_String_Contents is ex/imported.
Radio class	- MUIA_Radio_Active is ex/imported.
Cycle class	- MUIA_Cycle_Active is ex/imported.
List class	- MUIA_List_Active is /ex/imported.
Text class	- MUIA_Text_Contents is ex/imported.
Slider class	- MUIA_Slider_Level is ex/imported.
Area class	- MUIA_Selected is ex/imported (e.g. for Checkmark gadgets)
Menuitem class	- MUIA_Checked is ex/imported (V9).
Group class	- MUIA_Group_ActivePage is ex/imported (V8).

**INPUTS**

**name** - Name of the file you wish to save the settings to. Usually you won't need to think of a real name but instead use one of the magic cookies

- MUIV\_Application\_Save\_ENV or
- MUIV\_Application\_Save\_ENVARC.

This will save your application's settings somewhere in env:mui/ or envarc:mui/, you needn't worry about it.

### EXAMPLE

see the sample program "Settings.c"

### SEE ALSO

MUIM\_Application\_Load, MUIA\_ExportID

## 2.9 MUIM\_Application\_SetMenuCheck (V4) (OBSOLETE)

### SYNOPSIS

DoMethod(obj,MUIM\_Application\_SetMenuCheck,ULONG MenuID, LONG stat);

### FUNCTION

Set or clear the checkmark of a menu item. The application will ask its sub windows for menu items with the given id and set/clear all found entries.

### INPUTS

**MenuID** - the value you wrote into the UserData field of struct NewMenu.

**set** - TRUE to set checkmark, FALSE to clear

### SEE ALSO

MUIM\_Application\_GetMenuCheck, MUIA\_Application\_Menu,

## 2.10 MUIM\_Application\_SetMenuState (V4) (OBSOLETE)

### SYNOPSIS

DoMethod(obj,MUIM\_Application\_SetMenuState,ULONG MenuID, LONG stat);

### FUNCTION

Enable or disable a menu item. The application will ask its sub windows for menu items with the given id and enable/disable all found entries.

### INPUTS

**MenuID** - the value you wrote into the UserData field of struct NewMenu.

**set** - TRUE to enable item, FALSE to disable.

### SEE ALSO

MUIM\_Application\_GetMenuState, MUIA\_Application\_Menu,

## 2.11 MUIM\_Application\_ShowHelp (V4)

### SYNOPSIS

DoMethod(obj,MUIM\_Application\_ShowHelp,Object \*window, char \*name, char \*node, LONG line);

### FUNCTION

Show an AmigaGuide help file. The application will be put to sleep until the file is displayed.

Usually, you don't need to call this method directly. MUI comes with a sophisticated online help system, you just need to supply your gadgets with help nodes and everything will be handled automatically.

### INPUTS

**window** - (Object \*) - Help will appear on this windows screen. May be NULL.

**name** - (char \*) - name of the help file

**node** - (char \*) - name of a node in this help file

**line** - (char \*) - line number

### SEE ALSO

MUIA\_HelpFile, MUIA\_HelpNode, MUIA\_HelpLine

## 2.12 MUIA\_Application\_Active – (V4) [ISG], BOOL

### FUNCTION

This attribute reflects the state that the user adjusted with commodities Exchange. MUI itself doesn't pay any attention to it, this is up to you.

### SEE ALSO

MUIA\_Application\_Broker

## 2.13 MUIA\_Application\_Author – (V4) [I.G], STRPTR

### FUNCTION

Name of the applications author.

### EXAMPLE

see MUIA\_Application\_Title

### SEE ALSO

MUIA\_Application\_Title,  
MUIA\_Application\_Version,  
MUIA\_Application\_Base

MUIA\_Application\_Copyright,  
MUIA\_Application\_Description,

## 2.14 MUIA\_Application\_Base – (V4) [I.G], STRPTR

### FUNCTION

The basename for an application. This name is used for the builtin ARexx port and for some internal file management.

A basename must neither contain spaces nor any special characters such as `"/()#?*..."`.

When your program is a single task application ( i.e. `MUIA_Application_SingleTask` is `TRUE` ), the base name will be used without further modification.

Otherwise, it gets a `".1"`, `".2"`, etc. appended, depending on how many applications are already running. If you need to know the name of your ARexx port, you can query the base name attribute after the application is created.

### EXAMPLE

see `MUIA_Application_Title`

### SEE ALSO

`MUIA_Application_Title`, `MUIA_Application_Version`, `MUIA_Application_Author`, `MUIA_Application_Copyright`, `MUIA_Application_Description`

## 2.15 MUIA\_Application\_Broker – (V4) [..G], Broker \*

### FUNCTION

If you need to attach some additional commodities objects to your application (e.g. because you need lots of hotkeys), you can obtain a pointer to the applications Broker structure and add some commodities objects.

MUI will free the complete broker when the application is disposed, no need for you to free your objects yourself.

To receive input from your objects, you will also need to install a `MUIA_Application_BrokerHook`.

### NOTE

Unless you have set `MUIA_Application_RequiresCX`, you must be prepared to receive a `NULL` pointer. In this case, the commodities interface is not available, maybe because the user installed a light version of MUI.

### SEE ALSO

`MUIA_Application_BrokerHook`

## 2.16 MUIA\_Application\_BrokerHook – (V4) [ISG], struct Hook \*

### FUNCTION

You specify a pointer to hook structure. The function will be called whenever a commodities message arrives (between MUI's `GetMsg()` and `ReplyMsg()`).

You receive a pointer to the application object as object in `a2` and a pointer to commodities `CxMsg` message in `a1`.

**NOTE**

The commodities interface isn't available in the memory saving "light" version of MUI. Your hook will never be called in this case.

**SEE ALSO**

MUIA\_Application\_Broker

## 2.17 MUIA\_Application\_BrokerPort – (V6) [..G], struct MsgPort \*

**FUNCTION**

Get a pointer to the applications commodities message port. If you want to add own Hotkeys to your application, you need a message port. Instead of creating your own, you should better use this one.

**NOTE**

Unless you have set MUIA\_Application\_RequiresCX, you must be prepared to receive a NULL pointer. In this case, the commodities interface is not available, maybe because the user installed a light version of MUI.

**SEE ALSO**

MUIA\_Application\_BrokerHook

## 2.18 MUIA\_Application\_BrokerPri – (V6) [I.G], LONG

**FUNCTION**

Adjust the priority of an applications broker.

**SEE ALSO**

MUIA\_Application\_BrokerHook

## 2.19 MUIA\_Application\_Commands – (V4) [ISG], struct MUI\_Command \*

**FUNCTION**

This attribute allows an application to include its own set of ARexx commands. You specify a pointer to an array of MUI\_Command structures, which look like this:

```
struct MUI_Command
{
    char        *mc_Name;
    char        *mc_Template;
    LONG        mc_Parameters;
    struct Hook *mc_Hook;
    LONG        mc_Reserved[5];
};
```

**mc\_Name** contains the name of your command. Commands are not case sensitive.

**mc\_Template** is an argument template that follows the same rules as `dos.library/ReadArgs()`. It may be NULL, in which case your command doesn't need any parameters.

**mc\_Parameters** is the number of parameters specified in the template array.

**mc\_Hook** is a pointer to the callback hook defining the function to be called.

You may specify any number of `MUI_Command` structures, but you must terminate your array with a NULL field.

When a command shows up an applications ARexx port, MUI parses the arguments according to the given template and calls the hook with the application object as hook object in `a2` and a pointer to an array of longwords containing the parameters in `a1`.

The result code of your hook will be replied to ARexx as `rc`.

If you have some simple ARexx commands that just emulate some user action (e.g. clicking a button), you can use the magic cookie `MC_TEMPLATE_ID` for `mc_Template` and a return id value for `mc_Parameters`. In this case, MUI will do no argument parsing and instead simply return the specified id value on the next call to `MUIM_Application_Input`.

For more sophisticated possibilities in ARexx callback hooks, please refer to `MUIA_Application_RexxMsg` and `MUIA_Application_RexxString`.

#### EXAMPLE

```
static struct MUI_Command commands[] =
{
    { "rescan", MC_TEMPLATE_ID, ID_RESCAN, NULL    },
    { "select", "PATTERN/A"    , 1          , &selhook },
    { NULL     , NULL          , NULL    , NULL    }
};
```

#### SEE ALSO

`MUIA_Application_RexxMsg`, `MUIA_Application_RexxString`

### 2.20 MUIA\_Application\_Copyright – (V4) [I.G], STRPTR FUNCTION

A copyright string, containing the year and the company.

#### EXAMPLE

see `MUIA_Application_Title`

#### SEE ALSO

`MUIA_Application_Title`, `MUIA_Application_Version`, `MUIA_Application_Author`, `MUIA_Application_Description`, `MUIA_Application_Base`

### 2.21 MUIA\_Application\_Description – (V4) [I.G], STRPTR FUNCTION

Short description, about 40 characters. Shown e.g. in commodities exchange.



**EXAMPLE**

```
see MUIA_Application_Title
```

**SEE ALSO**

MUIA\_Application\_Title, MUIA\_Application\_Version, MUIA\_Application\_Author, MUIA\_Application\_Copyright, MUIA\_Application\_Base

## 2.22 MUIA\_Application\_DiskObject – (V4) [ISG], struct DiskObject \*

**FUNCTION**

Pointer to a struct DiskObject, e.g. obtained from GetDiskObject(). If present, MUI will use this object for the AppIcon when your application gets iconified.

Otherwise MUI will try to locate "env:sys/dev\_mui.info" and, if not present, fall back to a default icon.

**EXAMPLE**

```
...
MUIA_Application_DiskObject,
    dobj = GetDiskObject("PROGDIR:MyApp"),
...

/* note that you have to free dobj yourself! */
```

**NOTE**

Unless you have set MUIA\_Application\_RequiresIconification, this attribute might have no effect, maybe because the user installed a light version of MUI. You must be prepared to receive a NULL pointer when you try to read it!

**SEE ALSO**

MUIA\_Application\_Iconified

## 2.23 MUIA\_Application\_DoubleStart – (V4) [..G], BOOL

**FUNCTION**

This attribute is set automatically when the user tries to start a MUIA\_SingleTask application twice. You can react on this and take appropriate actions, e.g. pop up a requester or quit yourself.

**SEE ALSO**

MUIA\_Application\_SingleTask

## 2.24 MUIA\_Application\_DropObject – (V5) [IS.], Object \*

**FUNCTION**

If your application is iconified and the user drops icons onto the AppIcon, the object specified here will receive the AppMessage.

**SEE ALSO**

MUIA\_Window\_AppWindow, MUIM\_CallHook

**2.25 MUIA\_Application\_ForceQuit – (V8) [..G], BOOL****FUNCTION**

When your input loop receives a MUIV\_Application\_ReturnID\_Quit, you should query this attribute. In case its TRUE, your program should exit quietly without popping up any safety requesters or other stuff.

MUI will e.g. set this if the user issued a "QUIT FORCE" ARexx command to your application.

**2.26 MUIA\_Application\_HelpFile – (V8) [ISG], STRPTR****FUNCTION**

This attribute allows defining an AmigaGuide style file to be displayed when the user requests online help.

When the HELP button is pressed and the application defines a MUIA\_Application\_HelpFile, MUI tries to obtain MUIA\_HelpNode from the current object (the one under the mouse pointer). If MUIA\_HelpNode is not defined, MUI continues asking the parent object for this attribute (usually a group, but remember: the parent of a windows root object is the window itself, the parent of a window is the application).

When a non NULL MUIA\_HelpNode is found, the same procedure is applied to MUIA\_HelpLine. Then MUI puts the application to sleep and displays the file at the position specified with MUIA\_HelpNode and/or MUIA\_HelpLine.

This behaviour allows you to define one MUIA\_Application\_HelpFile for your application object and different help nodes and lines for your applications windows and/or gadgets.

**EXAMPLE**

```
ApplicationObject,
...
MUIA_Application_HelpFile, "progdire:myapp.guide",
...,
SubWindow, WindowObject,
    MUIA_Window_Title, "Prefs Window",
    ...,
    MUIA_HelpNode, "prefs-section",
    ...,
    End,

SubWindow, WindowObject,
    MUIA_Window_Title, "Play Window",
    ...,
    MUIA_HelpNode, "play-section",
    ...,
    WindowContents, VGroup,
    ...,
    Child, StringObject,
        MUIA_HelpNode, "play-string",
        ...,
```

```

        End,
    End,
End,
End;

```

In this case, the user will get the prefs-section chapter of "myapp.guide" when he requests help in the Prefs window, the play-string chapter when he requests help over the string gadget in the Play window or the play-section chapter somewhere else in the Play window.

## NOTE

Since muimaster.library V8, this attribute replaces the old and obsolete MUIA\_HelpFile attribute. MUI no longer supports the possibility to specify different help files for different parts of your application. This step was necessary due to some other internal changes and enhancements.

## SEE ALSO

MUIA\_HelpNode, MUIA\_HelpLine

## 2.27 MUIA\_Application\_Iconified – (V4) [.SG], BOOL

### FUNCTION

Setting this attribute to TRUE causes the application to become iconified. Every open window will be closed and a (configurable) AppIcon will appear on the workbench.

Same thing happens when the user hits the iconify gadget in the window border or uses commodities Exchange to hide your applications interface.

There is no way for you to prevent your application from being iconified. However, you can react on the iconification by listening to the MUIA\_Application\_Iconified attribute with notification. This allows you to free some resources you don't need in iconified state.

When an application is iconified and you try to open a window, the window won't open immediately. Instead MUI remembers this action and opens the window once the application is uniconified again.

## EXAMPLE

```

/* inform the main input loop of iconification events */

#define ID_HIDE 42
#define ID_SHOW 24

DoMethod(app,MUIM_Notify,
    MUIA_Application_Iconified, TRUE,
    app, 2, MUIM_Application_ReturnID, ID_HIDE);

DoMethod(app,MUIM_Notify,
    MUIA_Application_Iconified, FALSE,
    app, 2, MUIM_Application_ReturnID, ID_SHOW);

```

**SEE ALSO**

MUIA\_Application\_DiskObject

## 2.28 MUIA\_Application\_Menu – (V4) [I.G], struct NewMenu \* (OBSOLETE)

**FUNCTION**

Obsolete, use MUIA\_Application\_Menustrip instead.

**SEE ALSO**

MUIA\_Application\_Menustrip

## 2.29 MUIA\_Application\_MenuAction – (V4) [..G], ULONG

**FUNCTION**

Whenever a menu item is selected, this attribute will be set to the corresponding UserData field of the gadtools NewMenu structure. This allows reacting on menu items via broadcasting.

**SEE ALSO**

MUIA\_Application\_Menu, MUIA\_Application\_MenuAction

## 2.30 MUIA\_Application\_MenuHelp – (V4) [..G], ULONG

**FUNCTION**

Whenever a menu item is selected with the help key, this attribute will be set to the corresponding UserData field of the gadtools NewMenu structure. Together with MUIM\_Application\_ShowHelp this allows creation of menu help texts.

**SEE ALSO**

MUIA\_Application\_Menu, MUIA\_Application\_ShowHelp

## 2.31 MUIA\_Application\_Menustrip – (V8) [I..], Object \*

**FUNCTION**

Specify a menu strip object for the application. The object is treated as a child of the application and will be disposed when the application is disposed.

Menustrip objects defined for the application are used as menu for every window of the application, as long as the window doesn't define its private menu.

MUIA\_Application\_Menustrip replaces the old and obsolete MUIA\_Application\_Menu tag.

Usually, you will create the menu object with MUI's builtin object library from a gadtools NewMenu structure, but its also OK to define the menu tree "by hand" using the Family class.

### 2.32 MUIA\_Application\_RexxHook – (V7) [ISG], struct Hook \*

#### FUNCTION

When specified, MUI calls this hook whenever a rexx message arrives and MUI can't map it to a builtin or a programmer specified command. The hook will be called with a pointer to itself in A0, a pointer to the application object in A2 and a pointer to a struct RexxMsg in A1.

The return code from the hook is used as result code when replying the message, the secondary result can be set with MUIA\_Application\_RexxString.

#### SEE ALSO

MUIA\_Application\_Commands

### 2.33 MUIA\_Application\_RexxMsg – (V4) [..G], struct RxMsg \*

#### FUNCTION

Within an ARexx callback hook, you can obtain a pointer to the RexxMsg that came with the command. This allows you to use some ARexx support functions coming with amiga.lib

#### SEE ALSO

MUIA\_Application\_Commands, MUIA\_Application\_RexxString

### 2.34 MUIA\_Application\_RexxString – (V4) [.S.], STRPTR

#### FUNCTION

ARexx allows returning a string as result of a function call. This attribute allows setting the result string within an ARexx callback hook.

The string is temporarily copied.

#### SEE ALSO

MUIA\_Application\_Commands, MUIA\_Application\_RexxMsg

### 2.35 MUIA\_Application\_SingleTask – (V4) [I.], BOOL

#### FUNCTION

Boolean value to indicate whether or not your application is a single task program. When set to TRUE, MUI will refuse to create more than one application object.

In this case, the already running application gets its MUIA\_DoubleStart attribute set to TRUE. You can listen to this and take appropriate actions, e.g. pop up a requester.

Examples for single task applications are the system preferences program. It doesn't make sense for them to run more than once.

#### SEE ALSO

MUIA\_Application\_DoubleStart

### 2.36 MUIA\_Application\_Sleep – (V4) [.S.], BOOL

#### FUNCTION

This attribute can be used to put a whole application to sleep. All open windows get disabled and a busy pointer appears.

This attribute contains a nesting count, if you tell your application to sleep twice, you will have to tell it to wake up twice too.

If you need to do some time consuming actions, you always should set this attribute to inform the user that you are currently unable to handle input.

A sleeping application's windows cannot be resized.

#### EXAMPLE

```
set(app,MUIA_Application_Sleep,TRUE ); // go to bed
calc_fractals();
set(app,MUIA_Application_Sleep,FALSE); // wake up
```

#### SEE ALSO

MUIA.Window\_Sleep, MUIM.Application\_InputBuffered

### 2.37 MUIA\_Application\_Title – (V4) [I.G], STRPTR

#### FUNCTION

This tag defines the title of an application. The title is e.g. shown in Commodities Exchange or in the MUI preferences program.

An application title shall not contain any version information, just the pure title. Also, special characters such as ":( )#?\*..." are not allowed.

You should use a quiet long and unique name for your applications. Naming it "Viewer" or "Browser" is not a wise choice.

The length of the name must not exceed 30 characters!

#### EXAMPLE

```
ApplicationObject,
    MUIA_Application_Title      , "WbMan",
    MUIA_Application_Version    , "$VER: WbMan 0.24 (19.7.93)",
    MUIA_Application_Copyright  , "© 1993 by Klaus Melchior",
    MUIA_Application_Author     , "Klaus Melchior",
    MUIA_Application_Description, "Manages the WBStartup.",
    MUIA_Application_Base       , "WBMAN",
    ...
```

#### SEE ALSO

MUIA\_Application\_Version,  
MUIA\_Application\_Author,  
MUIA\_Application\_Base

MUIA\_Application\_Copyright,  
MUIA\_Application\_Description,

## 2.38 MUIA\_Application\_UseCommodities – (V10) [I.], BOOL

### FUNCTION

When set to FALSE, the application will run without a commodities interface. Think very well before using this tag!

### SEE ALSO

MUIA\_Application\_UseRexx

## 2.39 MUIA\_Application\_UseRexx – (V10) [I.], BOOL

### FUNCTION

When set to FALSE, the application will run without an ARexx interface. Think very well before using this tag!

### SEE ALSO

MUIA\_Application\_UseCommodities

## 2.40 MUIA\_Application\_Version – (V4) [I.G], STRPTR

### FUNCTION

Define a version string for an application. This string shall follow standard version string conventions but must **not** contain a leading "\0".

### EXAMPLE

see MUIA\_Application\_Title

### SEE ALSO

MUIA\_Application\_Title,  
MUIA\_Application\_Author,  
MUIA\_Application\_Base

MUIA\_Application\_Copyright,  
MUIA\_Application\_Description,

## 2.41 MUIA\_Application\_Window – (V4) [I.], Object \*

### FUNCTION

A pointer to a MUI object of Window class. An application may have any number of sub windows, each of them being a child of the application.

When the application receives some kind of user input through its IDCMP, it diverts the message down to its children, as long as they are opened.

Things like iconification or preferences changes cause the application object to temporarily close every open window (and reopen it later). Your main program normally doesn't need to deal with these things.

As with the children of group class, it's common to use a call to MUI\_NewObject() as value for this attribute. No error checking needs to be done, the application object handles every failure automatically.

When you dispose your application, its sub windows will also get deleted. Thus, the only thing to do to remove your application is a

```
MUI_DisposeObject(ApplicationObject);
```

Every window, every gadget, every memory will be freed by this single call.

#### **EXAMPLE**

Please refer to one of the example programs.

#### **SEE ALSO**

### **3 Area.mui**

Area class is a super class for every other MUI class except windows and applications. It holds information about an objects current position, size and weight and manages frames, fonts and backgrounds.

Additionally, area class handles the user input. By setting an objects `MUIA_InputMode`, you can make it behave like a button or like a toggle gadget. That's why MUI doesn't offer an extra button class. A button is simply a text object with a raised frame and a relverify input mode. Since especially group class is a subclass of area, you can create rather complex buttons consisting of many other display elements.

#### **3.1 MUIM\_AskMinMax (V4)**

[For use within custom classes only]

##### **SYNOPSIS**

```
DoMethod(obj,MUIM_AskMinMax,struct MUI_MinMax *MinMaxInfo);
```

##### **FUNCTION**

see developer documentation.

#### **3.2 MUIM\_Cleanup (V4)**

[For use within custom classes only]

##### **SYNOPSIS**

```
DoMethod(obj,MUIM_Cleanup,);
```

##### **FUNCTION**

see developer documentation.

#### **3.3 MUIM\_Draw (V4)**

[For use within custom classes only]

##### **SYNOPSIS**

```
DoMethod(obj,MUIM_Draw,ULONG flags);
```

##### **FUNCTION**

see developer documentation.



### 3.4 MUIM\_HandleInput (V4)

[For use within custom classes only]

#### SYNOPSIS

```
DoMethod(obj,MUIM_HandleInput,struct IntuiMessage *img, LONG muikey);
```

#### FUNCTION

see developer documentation.

### 3.5 MUIM\_Hide (V4)

[For use within custom classes only]

#### SYNOPSIS

```
DoMethod(obj,MUIM_Hide,);
```

#### FUNCTION

see developer documentation.

### 3.6 MUIM\_Setup (V4)

[For use within custom classes only]

#### SYNOPSIS

```
DoMethod(obj,MUIM_Setup,struct MUI_RenderInfo *RenderInfo);
```

#### FUNCTION

see developer documentation.

### 3.7 MUIM\_Show (V4)

[For use within custom classes only]

#### SYNOPSIS

```
DoMethod(obj,MUIM_Show,);
```

#### FUNCTION

see developer documentation.

### 3.8 MUIA\_ApplicationObject – (V4) [..G], Object \*

#### FUNCTION

You can obtain a pointer to the application object that some gadget belongs to by using this attribute. Useful mainly within callback hooks if you do not want to deal with global variables.

**SEE ALSO**

MUIA\_WindowObject

**3.9 MUIA\_Background – (V4) [IS.], LONG****FUNCTION**

Adjust the background for an object.

Every MUI object has its own background setting. The background is displayed "behind" the actual object contents, e.g. behind a the text of a text object or behind the image of an image object.

This attribute takes the same values as MUIA\_Image\_Spec, please refer to aut-docs of image class for a complete description.

An object without a specific background setting will inherit the pattern from its parent group. The default background for a window and many other background patterns are adjustable with the preferences program.

Only a few MUII\_XXXXXXX tags make sense as background. Important are:

**MUII\_ButtonBack:** You have to set this when you create a button gadget. Thus, your button will be displayed in the users preferred style.

**MUII\_TextBack:** Set this when you create a text object with a TextFrame, e.g. some kind of status line. Do *\*not\** use MUII\_TextBack for simple text without frame (e.g. gadget labels).

MUII\_BACKGROUND  
MUII\_SHADOW  
MUII\_SHINE  
MUII\_FILL  
MUII\_SHADOWBACK  
MUII\_SHADOWFILL  
MUII\_SHADOWSHINE  
MUII\_FILLBACK  
MUII\_FILLSHINE  
MUII\_SHINEBACK  
MUII\_SHINEBACK2

One of MUI's predefined pattern. These are not configurable by the user and will always look the same.

**NOTE**

It is **important** that you test your programs with a fancy pattern configuration. With the default setting you won't notice any errors in your backgrounds.

**3.10 MUIA\_BottomEdge – (V4) [..G], LONG****FUNCTION**

You can use this to read the current position and dimension of an object, if you e.g. need it to pop up some requester below.

Of course, this attribute is only valid when the parent window of the object is currently open.

**SEE ALSO**

MUIA\_TopEdge, MUIA\_Width, MUIA\_Height, MUIA\_RightEdge, MUIA\_LeftEdge

### 3.11 MUIA\_ControlChar – (V4) [I..], char

#### FUNCTION

Pressing the control char will have the same effect as pressing return if the object was active.

This can be used to create old style key shortcuts.

#### NOTE

Using an uppercase control char will force the user to press shift.

#### SEE ALSO

mui.h / KeyButton() macro

### 3.12 MUIA\_Disabled – (V4) [ISG], BOOL

#### FUNCTION

Disable or enable a gadget. Setting this attribute causes a gadget to become disabled, it gets a ghost pattern and doesn't respond to user input any longer.

Disabled gadgets cannot be activated with the TAB key.

Using MUIA\_Disable on a group of objects will disable all objects within that group.

#### EXAMPLE

```
/* we have a radio button gadget with three      */
/* entries, the third should enable a string gadget */
/* with additional parameters                      */
/*
DoMethod(radio, MUIM_Notify, MUIA_Radio_Active, 0,
    string, 3, MUIM_Set, MUIA_Disabled, TRUE);

DoMethod(radio, MUIM_Notify, MUIA_Radio_Active, 1,
    string, 3, MUIM_Set, MUIA_Disabled, TRUE);

DoMethod(radio, MUIM_Notify, MUIA_Radio_Active, 2,
    string, 3, MUIM_Set, MUIA_Disabled, FALSE);
*/
```

### 3.13 MUIA\_ExportID – (V4) [ISG], LONG

#### FUNCTION

Objects with a non NULL MUIA\_ExportID export their contents during MUIM\_Application.Save and import them during MUIM\_Application.Load.

You have to use different ExportIDs for your objects!

#### SEE ALSO

MUIM\_Application.Save, MUIM\_Application.Load

### 3.14 MUIA\_FixHeight – (V4) [I.], LONG

#### FUNCTION

Give your object a fixed pixel height. This tag is absolutely not needed in a general MUI application and only present for emergency situations. Please think twice before using it!

#### EXAMPLE

```
/* create an 8x8 pixel rectangle with FILLPEN */

RectangleObject,
    MUIA_FixWidth   , 8,
    MUIA_FixHeight  , 8,
    MUIA_Background, MUII_FILL,
    End;
```

#### SEE ALSO

MUIA\_FixWidth, MUIA\_FixWidthTxt, MUIA\_FixHeightTxt

### 3.15 MUIA\_FixHeightTxt – (V4) [I.], LONG

#### FUNCTION

Give your object a fixed pixel height. The height will match the height of the given string. This tag is absolutely not needed in a general MUI application and only present for emergency situations. Please think twice before using it!

#### EXAMPLE

```
/* create a fixed size rectangle with FILLPEN */

RectangleObject,
    MUIA_FixWidthTxt , "00:00:00",
    MUIA_FixHeightTxt, "\n\n",
    MUIA_Background  , MUII_FILL,
    End;
```

#### SEE ALSO

MUIA\_FixHeight, MUIA\_FixWidth, MUIA\_FixWidthTxt

### 3.16 MUIA\_FixWidth – (V4) [I.], LONG

#### FUNCTION

Give your object a fixed pixel width. This tag is absolutely not needed in a general MUI application and only present for emergency situations. Please think twice before using it!

**EXAMPLE**

```
/* create an 8x8 pixel rectangle with FILLPEN */

RectangleObject,
    MUIA_FixWidth   , 8,
    MUIA_FixHeight  , 8,
    MUIA_Background, MUII_FILL,
    End;
```

**SEE ALSO**

MUIA\_FixHeight, MUIA\_FixWidthTxt, MUIA\_FixHeightTxt

**3.17 MUIA\_FixWidthTxt – (V4) [I..], STRPTR****FUNCTION**

Give your object a fixed pixel width. The width will match the width of the given string. This tag is absolutely not needed in a general MUI application and only present for emergency situations. Please think twice before using it!

**EXAMPLE**

```
/* create a fixed size rectangle with FILLPEN */

RectangleObject,
    MUIA_FixWidthTxt , "00:00:00",
    MUIA_FixHeightTxt, "\n\n",
    MUIA_Background  , MUII_FILL,
    End;
```

**SEE ALSO**

MUIA\_FixHeight, MUIA\_FixWidth, MUIA\_FixHeightTxt

**3.18 MUIA\_Font – (V4) [I.G], struct TextFont \*****SPECIAL INPUTS**

MUIV\_Font\_Inherit  
 MUIV\_Font\_Normal  
 MUIV\_Font\_List  
 MUIV\_Font\_Tiny  
 MUIV\_Font\_Fixed  
 MUIV\_Font\_Title  
 MUIV\_Font\_Big

**FUNCTION**

Every MUI object can have its own font, just set it with this tag. Objects without an explicit font setting will inherit it from their parent group.

You normally won't need to open a font yourself, just use one of the predefined values to get a font from the users preferences.

**EXAMPLE**

```

/* since the text contains tabs,          */
/* use the fixed width font for displaying */

msgread = FloattextObject,
    MUIA_Font, MUIV_Font_Fixed,
    ...,
    End;

```

**3.19 MUIA\_Frame – (V4) [I.], LONG****SPECIAL INPUTS**

MUIV\_Frame\_None  
 MUIV\_Frame\_Button  
 MUIV\_Frame\_ImageButton  
 MUIV\_Frame\_Text  
 MUIV\_Frame\_String  
 MUIV\_Frame\_ReadList  
 MUIV\_Frame\_InputList  
 MUIV\_Frame\_Prop  
 MUIV\_Frame\_Gauge  
 MUIV\_Frame\_Group  
 MUIV\_Frame\_PopUp  
 MUIV\_Frame\_Virtual  
 MUIV\_Frame\_Slider  
 MUIV\_Frame\_Count

**FUNCTION**

Define a frame for the current object. Since area class is a superclass for all elements in a window, you can assign frames to every object you wish.

You don't adjust the style of your frame directly, instead you only specify a type:

**MUIV\_Frame\_Button** for standard buttons with text in it.

**MUIV\_Frame\_ImageButton** for small buttons with images, e.g. the arrows of a scrollbar.

**MUIV\_Frame\_Text** for a text field, e.g. a status line display.

**MUIV\_Frame\_String** for a string gadget.

**MUIV\_Frame\_ReadList** for a read only list.

**MUIV\_Frame\_InputList** for a list that handles input (has a cursor).

**MUIV\_Frame\_Prop** for proportional gadgets.

**MUIV\_Frame\_Group** for groups.

How the frame is going to look is adjustable via the preferences program.

Four spacing values belong to each frame that tell MUI how many pixels should be left free between the frame and its contents. These spacing values are also user adjustable as long as you don't override them with one of the MUIA\_InnerXXXX tags.

**NOTE**

The first object in a window (MUIA\_Window\_RootObject) may **not** have a frame. If you need this you will have to create a dummy group with just one child.

**EXAMPLE**

```
strobj = StringObject,
    MUIA_Frame, MUIV_Frame_String,
    End;
```

**SEE ALSO**

MUIA\_InnerLeft, MUIA\_InnerRight, MUIA\_InnerTop, MUIA\_InnerBottom

**3.20 MUIA\_FramePhantomHoriz – (V4) [I.], BOOL****FUNCTION**

Setting this to TRUE causes the specified frame to be a horizontal phantom frame. The frame will not appear but its vertical components (frame height, inner top and inner bottom spacing) will be used to calculate positions and dimensions (horizontal components are treated as 0).

This is extremely useful for a correct labeling of objects. You would e.g. label a string gadget by using a text object with a phantom string frame. Thus, the label text will be always on the same vertical position as the string gadget text, no matter what spacing values the user configured.

**SEE ALSO**

Label() macros in "mui.h".

**3.21 MUIA\_FrameTitle – (V4) [I.], STRPTR****FUNCTION**

This tag identifies a text string that will be displayed centered in the top line of a frame. This can become handy if you want to name groups of objects.

You may not use MUIA\_FrameTitle without defining a MUIA\_Frame.

**EXAMPLE**

```
VGroup,
    MUIA_Frame      , MUIV_Frame_Group,
    MUIA_FrameTitle, "Spacing",
    ...
```

**SEE ALSO**

MUIA\_Frame

### 3.22 MUIA\_Height – (V4) [..G], LONG

#### FUNCTION

You can use this to read the current position and dimension of an object, if you e.g. need it to pop up some requester below.

Of course, this attribute is only valid when the parent window of the object is currently open.

#### SEE ALSO

MUIA\_TopEdge, MUIA\_Width, MUIA\_LeftEdge, MUIA\_RightEdge,  
MUIA\_BottomEdge

### 3.23 MUIA\_HorizWeight – (V4) [I.], LONG

#### FUNCTION

Adjust the horizontal weight of an object. Usually you can simply use MUIA\_Weight instead of this tag but in some two-dimensional groups it may become handy to have different horizontal and vertical weights.

#### SEE ALSO

MUIA\_Weight

### 3.24 MUIA\_InnerBottom – (V4) [I.], LONG

#### FUNCTION

Adjust the space between an object and its frame. Usually you shouldn't use this tag since you will override the users preferred default setting.

#### SEE ALSO

MUIA\_Frame

### 3.25 MUIA\_InnerLeft – (V4) [I.], LONG

#### FUNCTION

Adjust the space between an object and its frame. Usually you shouldn't use this tag since you will override the users preferred default setting.

#### SEE ALSO

MUIA\_Frame

### 3.26 MUIA\_InnerRight – (V4) [I.], LONG

#### FUNCTION

Adjust the space between an object and its frame. Usually you shouldn't use this tag since you will override the users preferred default setting.

#### SEE ALSO

MUIA\_Frame



**3.27 MUIA\_InnerTop – (V4) [I.], LONG****FUNCTION**

Adjust the space between an object and its frame. Usually you shouldn't use this tag since you will override the users preferred default setting.

**SEE ALSO**

MUIA\_Frame

**3.28 MUIA\_InputMode – (V4) [I.], LONG****SPECIAL INPUTS**

MUIV\_InputMode\_None  
 MUIV\_InputMode\_RelVerify  
 MUIV\_InputMode\_Immediate  
 MUIV\_InputMode\_Toggle

**FUNCTION**

Adjust the input mode for an object.

MUI has no distinct button class. Instead you can make every object (even groups) behave like a button by setting an input mode for them. Several input modes are available:

**MUIV\_InputMode\_None:** No input, this is not a gadget.

**MUIV\_InputMode\_RelVerify:** For buttons and similar stuff.

**MUIV\_InputMode\_Immediate:** Used e.g. in a radio button object.

**MUIV\_InputMode\_Toggle:** For things like checkmark gadgets.

The input mode setting determines how a user action will trigger the attributes MUIA\_Selected, MUIA\_Pressed and MUIA\_Timer. See their documentation for details.

**EXAMPLE**

```
/* A traditional button, just a text object with */
/* a button frame and a relverify input mode:    */

okbutton = TextObject,
    MUIA_Frame      , MUIV_Frame_Button,
    MUIA_InputMode  , MUIV_InputMode_RelVerify,
    MUIA_Text_Contents, "OK",
    ...
```

**SEE ALSO**

MUIA\_Selected, MUIA\_Timer, MUIA\_Pressed

### 3.29 MUIA\_LeftEdge – (V4) [..G], LONG

#### FUNCTION

You can use this to read the current position and dimension of an object, if you e.g. need it to pop up some requester below.

Of course, this attribute is only valid when the parent window of the object is currently open.

#### SEE ALSO

MUIA\_TopEdge, MUIA\_Width, MUIA\_Height, MUIA\_RightEdge,  
MUIA\_BottomEdge

### 3.30 MUIA\_Pressed – (V4) [..G], BOOL

#### FUNCTION

Learn if a button is pressed (or released). The MUIA\_Pressed attribute of a gadget is triggered by some user action, depending on the input mode:

- MUIV\_InputMode\_RelVerify:
  - set when lmb is pressed.
  - cleared when lmb is released and the mouse is still over the gadget (otherwise it will be cleared too, but without triggering a notification event).
- MUIV\_InputMode\_Immediate:
  - undefined, use MUIA\_Selected for this.
- MUIV\_InputMode\_Toggle:
  - undefined, use MUIA\_Selected for this.

Waiting for MUIA\_Pressed getting FALSE is the usual way to react on button gadgets.

#### EXAMPLE

```
DoMethod(btcancel,MUIM_Notify,MUIA_Pressed,FALSE,
  app,2,MUIM_Application_ReturnID,ID_CANCEL);
```

#### SEE ALSO

MUIA\_Selected, MUIA\_Timer, MUIA\_ShowSelState, MUIA\_InputMode

### 3.31 MUIA\_RightEdge – (V4) [..G], LONG

#### FUNCTION

You can use this to read the current position and dimension of an object, if you e.g. need it to pop up some requester below.

Of course, this attribute is only valid when the parent window of the object is currently open.

**SEE ALSO**

MUIA\_TopEdge, MUIA\_Width, MUIA\_Height, MUIA\_LeftEdge,  
MUIA\_BottomEdge

**3.32 MUIA\_Selected – (V4) [ISG], BOOL****FUNCTION**

Get and set the selected state of a gadget. This attribute can be triggered by the user clicking on the gadget (or using the keyboard), depending on the input mode:

- MUIV\_InputMode\_RelVerify:
  - set when lmb is pressed.
  - cleared when lmb is released.
  - cleared when the gadget is selected and the mouse leaves the gadget box.
  - set when the mouse reenters the gadget box.
- MUIV\_InputMode\_Immediate:
  - set when lmb is pressed.
- MUIV\_InputMode\_Toggle:
  - toggled when lmb is pressed.

Of course you may set this attribute yourself, e.g. to adjust the state of a check-mark gadget.

A selected gadget will display its border reverse and get the configured MUI\_SelectedBack background. This can be avoided using the MUIA\_ShowSelState tag.

**SEE ALSO**

MUIA\_Pressed, MUIA\_Timer, MUIA\_ShowSelState, MUIA\_InputMode

**3.33 MUIA\_ShowMe – (V4) [ISG], BOOL****FUNCTION**

Objects with this attribute set are not displayed. You can set MUIA\_ShowMe at any time, causing objects to appear and to disappear immediately. A new layout is calculated whenever some objects are shown or hidden. When necessary, MUI will resize the parent window to make place for the new objects.

**NOTE**

Currently, MUI does a complete window refresh after showing/hiding objects. This behaviour might get improved in the future.

**3.34 MUIA\_ShowSelState – (V4) [I.], BOOL****FUNCTION**

Normally a gadget will reverse its frame and display the configured MUI\_SelectedBack background pattern in its selected state. For some objects (e.g. checkmarks) this is not recommended and can be suppressed by setting MUIA\_ShowSelState to FALSE.

**SEE ALSO**

MUIA\_Selected

**3.35 MUIA\_Timer – (V4) [..G], LONG****FUNCTION**

MUIA\_Timer gets triggered when a relverify button is pressed and (after a little delay) increases every INTUITICK as long as the mouse remains over the gadget.

This makes it possible to have buttons repeatedly cause some actions, just like the arrow gadgets of a scrollbar.

**EXAMPLE**

```
DoMethod(btmore,MUIM_Notify,MUIA_Timer,MUIV_EveryTime,
    app,2,MUIM_Application_ReturnID,ID_MORE);
```

```
DoMethod(btless,MUIM_Notify,MUIA_Timer,MUIV_EveryTime,
    app,2,MUIM_Application_ReturnID,ID_LESS);
```

**SEE ALSO**

MUIA\_Pressed, MUIA\_Selected

**3.36 MUIA\_TopEdge – (V4) [..G], LONG****FUNCTION**

You can use this to read the current position and dimension of an object, if you e.g. need it to pop up some requester below.

Of course, this attribute is only valid when the parent window of the object is currently open.

**SEE ALSO**

MUIA\_LeftEdge,      MUIA\_Width,      MUIA\_Height,      MUIA\_RightEdge,  
MUIA\_BottomEdge

**3.37 MUIA\_VertWeight – (V4) [I.], LONG****FUNCTION**

Adjust the vertical weight of an object. Usually you can simply use MUIA\_Weight instead of this tag but in some two-dimensional groups it may become handy to have different horizontal and vertical weights.

**SEE ALSO**

MUIA\_Weight

### 3.38 MUIA\_Weight – (V4) [I.], LONG

#### FUNCTION

This tag is a shorthand for MUIA\_HorizWeight and MUIA\_VertHeight, it sets both weights at once.

The weight of an object determines how much room it will get during the layout process. Imagine you have a 100 pixel wide horizontal group with two string gadgets. Usually, each gadget will get half of the room and be 50 pixels wide. If you feel the left gadget is more important and should be bigger, you can give it a weight of 200 (and 100 for the right gadget). Because the left gadget is twice as "heavy" as the right gadget, it will become twice as big (about 66 pixel) as the right one (34 pixel).

Of course giving weights only makes sense if the object is resizable. A MUIA\_VertWeight for a (always fixed height) string gadget is useless.

An object with a weight of 0 will always stay at its minimum size.

By default, all objects have a weight of 100.

#### EXAMPLE

```
HGroup,
  StringGadget, MUIA_Weight, 50, End,
  StringGadget, MUIA_Weight, 100, End,
  StringGadget, MUIA_Weight, 200, End,
End;
```

#### SEE ALSO

MUIA\_HorizWeight, MUIA\_VertWeight

### 3.39 MUIA\_Width – (V4) [..G], LONG

#### FUNCTION

You can use this to read the current position and dimension of an object, if you e.g. need it to pop up some requester below.

Of course, this attribute is only valid when the parent window of the object is currently open.

#### SEE ALSO

MUIA\_TopEdge, MUIA\_LeftEdge, MUIA\_Height, MUIA\_RightEdge,  
MUIA\_BottomEdge

### 3.40 MUIA\_Window – (V4) [..G], struct Window \*

#### FUNCTION

This attribute can be used to get a pointer to the intuition window structure of the parent window of the object. This pointer could e.g. be used in calls to asl.library.

The result is only valid when the window is opened.

#### SEE ALSO

MUIA\_Window\_Window

### 3.41 MUIA\_WindowObject – (V4) [..G], Object \*

#### FUNCTION

You can obtain a pointer to the window object that some gadget belongs to by using this attribute. Useful mainly within callback hooks if you do not want to deal with global variables.

#### SEE ALSO

MUIA\_ApplicationObject

## 4 Bitmap.mui

The Bitmap class allows including self-made image data in MUI applications. Usually, image class was intended to be used for this purpose but unfortunately, its design was not very useful.

In its most simple usage, Bitmap class just display a given BitMap. However, you can also tell it to do automatic color remapping to match the current display context and you can define a transparent color to make the BitMap appear on any background.

### 4.1 MUIA\_Bitmap\_Bitmap – (V8) [ISG], struct BitMap \*

#### FUNCTION

This attribute specifies a pointer to a struct BitMap. Note that specifying only a BitMap isn't enough, you have to tell MUI about the pixel width and height with MUIA\_Bitmap\_Width and MUIA\_Bitmap\_Height too.

#### SEE ALSO

MUIA\_Bitmap\_Width, MUIA\_Bitmap\_Height, MUIA\_Bitmap\_Transparent, MUIA\_Bitmap\_SourceColors, MUIA\_Bitmap\_MappingTable

### 4.2 MUIA\_Bitmap\_Height – (V8) [ISG], LONG

#### FUNCTION

Define the pixel height of the BitMap.

#### NOTE

By default, the bitmap object has a minimum size of 1 pixel and an unlimited maximum size. If the space is too small to hold your BitMap, it will be clipped. Usually, you will use MUIA\_FixWidth and MUIA\_FixHeight with BitMap objects to make them always exactly as big as the bitmap.

#### SEE ALSO

MUIA\_Bitmap\_Bitmap, MUIA\_Bitmap\_Width, MUIA\_Bitmap\_Transparent, MUIA\_Bitmap\_SourceColors, MUIA\_Bitmap\_MappingTable

### 4.3 MUIA\_Bitmap\_MappingTable – (V8) [ISG], UBYTE \*

#### FUNCTION

Address of an array of UBYTES, one for each color of the source BitMap. MUI will remap the BitMap according to the contents of the array.

Since MUI applications usually don't know about their display environment, this tag is rarely used. Instead, MUIA\_Bitmap\_SourceColors can be used to allow context sensitive color remapping.

#### SEE ALSO

MUIA\_Bitmap\_Bitmap, MUIA\_Bitmap\_Height, MUIA\_Bitmap\_Width,  
MUIA\_Bitmap\_Transparent

### 4.4 MUIA\_Bitmap\_SourceColors – (V8) [ISG], ULONG \*

#### FUNCTION

This attribute defines the color palette of the source BitMap. If specified, MUI will try to locate these colors on the current screen and remap the BitMap accordingly.

You can e.g. specify some great looking 8-color images for several buttons of your application and MUI will ensure they look fine even on 4-color screens or on screens with completely different colors.

When running Kickstart V39 or higher, MUI will use ObtainBestPen() to find or create your colors. Below V39, a simple color-map search is performed to find the best matching entry, but no colors will be changed.

The source palette is specified with an array of ULONGs, three entries per color, 32bits per gun.

#### EXAMPLE

```
/* MagicWB-like palette for an 8-color image */
```

```
const ULONG aboutlogo_colors[24] =
{
    0xaaaaaaaa,0xaaaaaaaa,0xa0a0a0a0,
    0x00000000,0x00000000,0x00000000,
    0xffffffff,0xffffffff,0xffffffff,
    0x66666666,0x88888888,0xbbbbbbbb,
    0x99999999,0x99999999,0x99999999,
    0xbbbbbbbb,0xbbbbbbbb,0xbbbbbbbb,
    0xbbbbbbbb,0xaaaaaaaa,0x99999999,
    0xffffffff,0xbbbbbbbb,0xaaaaaaaa
};
```

#### SEE ALSO

MUIA\_Bitmap\_Bitmap, MUIA\_Bitmap\_Height, MUIA\_Bitmap\_Width,  
MUIA\_Bitmap\_MappingTable, MUIA\_Bitmap\_MappingTable

### 4.5 MUIA\_Bitmap\_Transparent – (V8) [ISG], LONG

#### FUNCTION

If specified, MUI will consider this color of the BitMap to be transparent. A mask plane will be generated and used for blitting, the background will shine through.

#### SEE ALSO

MUIA\_Bitmap\_Bitmap, MUIA\_Bitmap\_Height, MUIA\_Bitmap\_Width,  
MUIA\_Bitmap\_SourceColors, MUIA\_Bitmap\_MappingTable

### 4.6 MUIA\_Bitmap\_Width – (V8) [ISG], LONG

#### FUNCTION

Define the pixel width of the BitMap.

#### NOTE

By default, the bitmap object has a minimum size of 1 pixel and an unlimited maximum size. If the space is too small to hold your BitMap, it will be clipped. Usually, you will use MUIA\_FixWidth and MUIA\_FixHeight with BitMap objects to make them always exactly as big as the bitmap.

#### SEE ALSO

MUIA\_Bitmap\_Bitmap, MUIA\_Bitmap\_Height, MUIA\_Bitmap\_Transparent,  
MUIA\_Bitmap\_SourceColors, MUIA\_Bitmap\_MappingTable

## 5 Bodychunk.mui

Big and colorful images (e.g. About-Logos) usually take lots of space when stored in a traditional BitMap structure. To save memory, you can decide to have the picture compressed in your code and use the Bodychunk class instead of the Bitmap class for displaying. MUI will then automatically decompress your image when its about to appear in a window.

Since Bodychunk class is a subclass of Bitmap class, you can of course use all the Bitmaps remapping and transparency features.

### 5.1 MUIA\_Bodychunk\_Body – (V8) [ISG], UBYTE \*

#### FUNCTION

Specify a pointer to the BODY data of your picture. This BODY data must follow normal IFF/ILBM conventions.

You have to supply MUIA\_Bitmap\_Width, MUIA\_Bitmap\_Height and MUIA\_Bodychunk\_Depth to describe the contents of the BODY data, otherwise MUI will fail to decompress it.

#### SEE ALSO

MUIA\_Bodychunk\_Depth, MUIA\_Bodychunk\_Compression,  
MUIA\_Bodychunk\_Masking



## 5.2 MUIA\_Bodychunk\_Compression – (V8) [ISG], UBYTE FUNCTION

MUI is able to uncompress byte&run compressed BODY chunks automatically. If your data is compressed, you must supply a value of cmpByteRun1 (==1) for this tag. Other compression techniques are not supported.

Omitting this tag or setting it to 0 indicates that the BODY data is uncompressed. Using the Bodychunk class doesn't make much sense in this case since its main purpose is to save memory for big images.

### SEE ALSO

MUIA\_Bodychunk\_Masking, MUIA\_Bodychunk\_Body

## 5.3 MUIA\_Bodychunk\_Depth – (V8) [ISG], LONG FUNCTION

Specify the depth of your picture here. This tag is required for correct BODY chunk parsing. Also remember to use MUIA\_Bodychunk\_Masking if your BODY data contains a masking bitplane.

### SEE ALSO

MUIA\_Bodychunk\_Body, MUIA\_Bodychunk\_Masking

## 5.4 MUIA\_Bodychunk\_Masking – (V8) [ISG], UBYTE FUNCTION

You must indicate if your BODY data contains a masking plane. Currently, MUI does not use this masking plane for any purpose, but this attribute is required to allow correct parsing of the BODY data.

### SEE ALSO

MUIA\_Bodychunk\_Body, MUIA\_Bodychunk\_Compression

## 6 Boopsi.mui

MUI's boopsi class provides an interface to standard, system style boopsi gadgets. Since boopsi gadgetclass misses some important features needed for an automatic layout system like MUI, there are several problems with such an interface. MUI tries to solve these problems with some additional attributes.

Coming with release 3.x of the amiga operating system are some very nice boopsi gadgets such as "colorwheel.gadget" or "gradientslider.gadget". With MUI's boopsi class, you can use these gadgets just as if they were MUI objects.

You can talk to a MUIized boopsi object as if it was the boopsi object itself. MUI will pass through all attributes and try to be completely transparent. Additionally, if a boopsi object generates notification events via IDCMP\_UPDATE, MUI turns them into MUI notification events. Thus, you can e.g. react on the change of WHEEL\_Saturation in a MUI colorwheel boopsi gadget as on any other MUI attribute.

An example program "BoopsiDoor.c" is provided to show how this magic works.

**NOTE**

OS 3.0/3.1 colorwheel.gadget can accidentally render itself one pixel too big, overwriting other parts of the window. As a workaround, MUI will subtract one from the width/height before passing it on to a colorwheel boopsi object.

**6.1 MUIA\_Boopsi\_Class – (V4) [ISG], struct IClass \*****FUNCTION**

Pointer to the (private) class you want to create a boopsi object from. Only useful if you previously generated your own boopsi class with MakeClass().

Of course you may not free the class until you're done with your object.

**SEE ALSO**

MUIA\_Boopsi\_ClassID

**6.2 MUIA\_Boopsi\_ClassID – (V4) [ISG], char \*****FUNCTION**

MUIA\_Boopsi\_ClassID specifies the name for the public Boopsi class you want to create an object of. It will only be used when MUIA\_Boopsi\_Class is NULL.

The public class must be in memory before you can create an instance of it, you will have to open the required class library by hand.

Note the string given to MUIA\_Boopsi\_ClassID must remain valid until you're done with the object.

**EXAMPLE**

```
/* Complete example code can be found in BoopsiDoor.c */

cwbase = OpenLibrary("gadgets/colorwheel.gadget",0);

Wheel = BoopsiObject, /* MUI and Boopsi tags mixed */
    NeXTFrame,
    MUIA_Boopsi_ClassID , "colorwheel.gadget",
    MUIA_Boopsi_MinWidth , 30, /* boopsi objects don't know */
    MUIA_Boopsi_MinHeight, 30, /* their sizes, so we help */
    MUIA_Boopsi_Remember , WHEEL_Saturation, /* keep important values */
    MUIA_Boopsi_Remember , WHEEL_Hue, /* during window resize */
    MUIA_Boopsi_TagScreen, WHEEL_Screen, /* this magic fills in */
    WHEEL_Screen , NULL, /* the screen pointer */
    GA_Left , 0,
    GA_Top , 0, /* MUI will automatically */
    GA_Width , 0, /* fill in the correct values */
    GA_Height , 0,
    ICA_TARGET , ICTARGET_IDCMP, /* needed for notification */
    End;

...

MUI_DisposeObject(wheel);
CloseLibrary(cwbase);
```

**SEE ALSO**

MUIA\_Boopsi\_Class

**6.3 MUIA\_Boopsi\_MaxHeight – (V4) [ISG], ULONG****FUNCTION**

For MUI's automatic layout system, it's required that objects know their minimum and maximums sizes. Since boopsi gadgets don't support this feature, you will have to help MUI and adjust these values by hand.

**Defaults**

MUIA_MinWidth	-	1 pixel
MUIA_MinHeight	-	1 pixel
MUIA_MaxWidth	-	unlimited
MUIA_MaxHeight	-	unlimited

**EXAMPLE**

see MUIA\_Boopsi\_ClassID

**SEE ALSO**

MUIA\_Boopsi\_ClassID

**6.4 MUIA\_Boopsi\_MaxWidth – (V4) [ISG], ULONG****FUNCTION**

For MUI's automatic layout system, it's required that objects know their minimum and maximums sizes. Since boopsi gadgets don't support this feature, you will have to help MUI and adjust these values by hand.

**Defaults**

MUIA_MinWidth	-	1 pixel
MUIA_MinHeight	-	1 pixel
MUIA_MaxWidth	-	unlimited
MUIA_MaxHeight	-	unlimited

**EXAMPLE**

see MUIA\_Boopsi\_ClassID

**SEE ALSO**

MUIA\_Boopsi\_ClassID

**6.5 MUIA\_Boopsi\_MinHeight – (V4) [ISG], ULONG****FUNCTION**

For MUI's automatic layout system, it's required that objects know their minimum and maximums sizes. Since boopsi gadgets don't support this feature, you will have to help MUI and adjust these values by hand.

**Defaults**

MUIA_MinWidth	-	1 pixel
MUIA_MinHeight	-	1 pixel
MUIA_MaxWidth	-	unlimited
MUIA_MaxHeight	-	unlimited

**EXAMPLE**

see MUIA\_Boopsi\_ClassID

**SEE ALSO**

MUIA\_Boopsi\_ClassID

**6.6 MUIA\_Boopsi\_MinWidth – (V4) [ISG], ULONG****FUNCTION**

For MUI's automatic layout system, it's required that objects know their minimum and maximums sizes. Since boopsi gadgets don't support this feature, you will have to help MUI and adjust these values by hand.

**Defaults**

MUIA_MinWidth	-	1 pixel
MUIA_MinHeight	-	1 pixel
MUIA_MaxWidth	-	unlimited
MUIA_MaxHeight	-	unlimited

**EXAMPLE**

see MUIA\_Boopsi\_ClassID

**SEE ALSO**

MUIA\_Boopsi\_ClassID

**6.7 MUIA\_Boopsi\_Object – (V4) [..G], Object \*****FUNCTION**

No input, just an output since this attribute is only gettable. What MUI returns when generating a BoopsiObject is a standard MUI object, not a pointer to the Boopsi gadget itself. In case you really need this Boopsi gadget pointer, you can obtain it by getting MUIA\_Boopsi\_Object from the MUI object.

Since MUI passes along every unknown attribute to the boopsi gadget, there should be no need for this tag anyway.

Note that the boopsi object pointer is only valid when the window is open!

**SEE ALSO**

MUIA\_Boopsi\_Class, MUIA\_Boopsi\_ClassID

## 6.8 MUIA\_Boopsi\_Remember – (V4) [I.], ULONG

### FUNCTION

Most boopsi objects are kind of silly, they don't support automatic resizing or jumping from screen to screen. Therefore, MUI sometimes needs to dispose and regenerate a boopsi object. This will result in losing the current state of the object, e.g. saturation and hue values in a colorwheel.

To solve this problem, you can tell MUI what attributes must be remembered during dispose/regeneration. For a colorwheel, this would e.g. be WHEEL\_Saturation and WHEEL\_Hue.

Before disposing the boopsi object, the remember tags are read and stored in a private buffer. After regeneration, the contents of this buffer are passed back to the boopsi again.

Note that you can define up to five MUIA\_Remember tags.

### BUGS

The remember procedure will not work when the attributes you want to remember are just pointers to data stored somewhere in the boopsi object.

### EXAMPLE

see MUIA\_Boopsi\_ClassID

### SEE ALSO

MUIA\_Boopsi\_ClassID

## 6.9 MUIA\_Boopsi\_Smart – (V9) [I.], BOOL

### FUNCTION

Specify TRUE for smart BOOPSI gadgets that allow resizing, e.g. the textfield.class. In this case, MUI will not dispose and recreate the object.

## 6.10 MUIA\_Boopsi\_TagDrawInfo – (V4) [ISG], ULONG

### FUNCTION

Unfortunately, most boopsi gadgets need information on the display environment they will reside in at object creation time. Due to MUI's concept, this information is not available that early.

To solve this problem, MUI doesn't generate the boopsi object instantly, creation is delayed until the window containing the gadget is opened.

At this time, MUI fills some values about display environment into the boopsi objects creation tag list. You have to tell MUI, what tags are actually needed.

With MUIA\_Boopsi\_TagDrawInfo you can tell MUI where to fill in a needed DrawInfo structure.

### EXAMPLE

If your boopsi gadget needs a pointer to a DrawInfo structure supplied with the MYBOOPSI\_DrawInfo tag, you would have to specify

```

BoopsiObject,
    RecessedFrame,
    ...
    MUIA_Boopsi_TagDrawInfo, MYBOOPSI_DrawInfo,
    ...
    MYBOOPSI_DrawInfo, 0, /* will be filled later by MUI */
    ...
    GA_Left , 0, /* needs to be there, will */
    GA_Top , 0, /* be filled later by MUI */
    GA_Width , 0,
    GA_Height, 0,

End;

```

### SEE ALSO

MUIA\_Boopsi\_ClassID, MUIA\_Boopsi\_TagScreen, MUIA\_Boopsi\_TagWindow

## 6.11 MUIA\_Boopsi\_TagScreen – (V4) [ISG], ULONG

### FUNCTION

Unfortunately, most boopsi gadgets need information on the display environment they will reside in at object creation time. Due to MUI's concept, this information is not available that early.

To solve this problem, MUI doesn't generate the boopsi object instantly, creation is delayed until the window containing the gadget is opened.

At this time, MUI fills some values about display environment into the boopsi objects creation tag list. You have to tell MUI, what tags are actually needed.

With MUIA\_Boopsi\_TagScreen you can tell MUI where to fill in a needed Screen structure.

### EXAMPLE

If your boopsi gadget needs a pointer to a Screen structure supplied with the MY-BOOPSI\_Screen tag, you would have to specify

```

BoopsiObject,
    RecessedFrame,
    ...
    MUIA_Boopsi_TagScreen, MYBOOPSI_Screen,
    ...
    MYBOOPSI_Screen, 0, /* will be filled later by MUI */
    ...
    GA_Left , 0, /* needs to be there, will */
    GA_Top , 0, /* be filled later by MUI */
    GA_Width , 0,
    GA_Height, 0,

End;

```

### SEE ALSO

MUIA\_Boopsi\_ClassID, MUIA\_Boopsi\_TagDrawInfo, MUIA\_Boopsi\_TagWindow

## 6.12 MUIA\_Boopsi\_TagWindow – (V4) [ISG], ULONG

### FUNCTION

Unfortunately, most boopsi gadgets need information on the display environment they will reside in at object creation time. Due to MUI's concept, this information is not available that early.

To solve this problem, MUI doesn't generate the boopsi object instantly, creation is delayed until the window containing the gadget is opened.

At this time, MUI fills some values about display environment into the boopsi objects creation tag list. You have to tell MUI, what tags are actually needed.

With MUIA\_Boopsi\_TagWindow you can tell MUI where to fill in a needed Window structure.

### EXAMPLE

If your boopsi gadget needs a pointer to a Window structure supplied with the MYBOOPSI\_Window tag, you would have to specify

```
BoopsiObject,
    RecessedFrame,
    ...
    MUIA_Boopsi_TagWindow, MYBOOPSI_Window,
    ...
    MYBOOPSI_Window, 0, /* will be filled later by MUI */
    ...
    GA_Left  , 0, /* needs to be there, will */
    GA_Top   , 0, /* be filled later by MUI */
    GA_Width , 0,
    GA_Height, 0,

    End;
```

### SEE ALSO

MUIA\_Boopsi\_ClassID, MUIA\_Boopsi\_TagDrawInfo, MUIA\_Boopsi\_TagWindow

## 7 Coloradjust.mui

Coloradjust class creates some gadgets that allow adjusting a single color. Depending on the operating system, different kinds of gadgets are be used. Kickstart 2.x users might only receive an RGB slider triple, Kickstart 3.x users could get an additional colorwheel if available. However, the outfit of this class is not important for you as a programmer.

### 7.1 MUIA\_Coloradjust\_Blue – (V4) [ISG], ULONG

#### FUNCTION

Set or get the 32-bit blue component of the adjusted color. Values range from 0 (no blue) to \$ffffff (full blue).

#### SEE ALSO

MUIA\_Coloradjust\_Green, MUIA\_Coloradjust\_Red, MUIA\_Coloradjust\_RGB, MUIA\_Coloradjust\_ModeID

## 7.2 MUIA\_Coloradjust\_Green – (V4) [ISG], ULONG

### FUNCTION

Set or get the 32-bit green component of the adjusted color. Values range from 0 (no green) to \$ffffff (full green).

### SEE ALSO

MUIA\_Coloradjust\_Red, MUIA\_Coloradjust\_Blue, MUIA\_Coloradjust\_RGB,  
MUIA\_Coloradjust\_ModeID

## 7.3 MUIA\_Coloradjust\_ModeID – (V4) [ISG], ULONG

### FUNCTION

This attribute tells the coloradjust object for which screen mode the color shall be adjusted. The object queries the display data base for some mode attributes (such as supported number of red/green/blue bits) and adjusts its display accordingly, giving the user an idea of what colors are supported.

Omitting this attribute does not affect the functionality of a coloradjust object. The user will still be able to adjust a color. However, if you know the ModeID, you should supply it.

### SEE ALSO

MUIA\_Coloradjust\_RGB

### EXAMPLE

```
set(cadj,MUIA_Coloradjust_ModeID,GetVPModeID(viewport));
```

## 7.4 MUIA\_Coloradjust\_Red – (V4) [ISG], ULONG

### FUNCTION

Set or get the 32-bit red component of the adjusted color. Values range from 0 (no red) to \$ffffff (full red).

### SEE ALSO

MUIA\_Coloradjust\_Green, MUIA\_Coloradjust\_Blue, MUIA\_Coloradjust\_RGB,  
MUIA\_Coloradjust\_ModeID

## 7.5 MUIA\_Coloradjust\_RGB – (V4) [ISG], ULONG \*

### FUNCTION

Set or get the red/green/blue values all at once. You pass in / receive a pointer to three longwords containing the 32-bit red, green and blue values.



**EXAMPLE**

```

ULONG rgb[3] = { 0xa000000,0xdeadbeaf,0x42424242 };
set(cadj,MUIA_Coloradjust_RGB,rgb);

ULONG *rgb;
get(cadj,MUIA_Coloradjust_RGB,&rgb);
printf("red=%08lx green=%08lx blue=%08lx\n",rgb[0],rgb[1],rgb[2]);

```

**SEE ALSO**

MUIA\_Coloradjust\_Green, MUIA\_Coloradjust\_Blue, MUIA\_Coloradjust\_Red,  
MUIA\_Coloradjust\_ModeID

**8 Colorfield.mui**

Colorfield class creates a rectangle filled with a specific color, useful e.g. within a palette requester. You can change the color of the field at any time by setting its RGB attributes.

The field will try to obtain an exclusive pen on the current screen. When none is available, it just displays some kind of rastered background. Maybe it will get a little more intelligent and try to display the color by mixing together some other colors, but thats a future topic.

Needless to say that Colorfield only works with Kickstart 3.x and above, since lower operating systems don't support pen sharing. When using this class with a lower OS, you will also get some kind of (boring) raster.

**8.1 MUIA\_Colorfield\_Blue – (V4) [ISG], ULONG****FUNCTION**

Set or get the 32-bit blue component of the fields color. Values range from 0 (no blue) to \$ffffff (full blue).

**SEE ALSO**

MUIA\_Colorfield\_Green, MUIA\_Colorfield\_Red, MUIA\_Colorfield\_RGB

**8.2 MUIA\_Colorfield\_Green – (V4) [ISG], ULONG****FUNCTION**

Set or get the 32-bit green component of the fields color. Values range from 0 (no green) to \$ffffff (full green).

**SEE ALSO**

MUIA\_Colorfield\_Red, MUIA\_Colorfield\_Blue, MUIA\_Colorfield\_RGB

**8.3 MUIA\_Colorfield\_Pen – (V4) [..G], ULONG****FUNCTION**

When specified, the colorfield uses exactly this pen instead of trying to obtain a new one.

**SEE ALSO**

MUIA\_Colorfield\_RGB

**8.4 MUIA\_Colorfield\_Red – (V4) [ISG], ULONG****FUNCTION**

Set or get the 32-bit red component of the fields color. Values range from 0 (no red) to \$ffffff (full red).

**SEE ALSO**

MUIA\_Colorfield\_Green, MUIA\_Colorfield\_Blue, MUIA\_Colorfield\_RGB

**8.5 MUIA\_Colorfield\_RGB – (V4) [ISG], ULONG \*****FUNCTION**

Set or get the red/green/blue values of a colorfield all at once. You pass in / receive a pointer to three longwords containing the 32-bit red, green and blue values.

**EXAMPLE**

```
ULONG rgb[3] = { 0xa000000,0xdeadbeaf,0x42424242 };
set(field,MUIA_Colorfield_RGB,rgb);

ULONG *rgb;
get(field,MUIA_Colorfield_RGB,&rgb);
printf("red=%08lx green=%08lx blue=%08lx\n",rgb[0],rgb[1],rgb[2]);
```

**SEE ALSO**

MUIA\_Colorfield\_Green, MUIA\_Colorfield\_Blue, MUIA\_Colorfield\_Red

**9 Colorpanel.mui**

This class is for preferences programs use only and currently not documented.

**10 Cycle.mui**

Cycle class generates the well known cycle gadgets. However, MUI cycle gadgets feature a (configurable) popup menu to avoid clicking through many entries.

**10.1 MUIA\_Cycle\_Active – (V4) [ISG], LONG**

SPECIAL INPUTS MUIV\_Cycle\_Active\_Next MUIV\_Cycle\_Active\_Prev

**FUNCTION**

This attribute defines the number of the active entry in the cycle gadgets. Valid range is from 0 for the first entry to NumEntries-1 for the last.

Setting MUIA\_Cycle\_Active causes the gadget to be updated. On the other hand, when the user plays around with the gadget, MUIA\_Cycle\_Active will always reflect the current state.

Using MUIV\_Cycle\_Active\_Next and MUIV\_Cycle\_Active\_Prev as attribute value during set causes the gadget to cycle through its entries in the given direction.

**EXAMPLE**

```
set(cycleobj,MUIA_Cycle_Active,3);
```

**SEE ALSO**

MUIA\_Cycle\_Entries

**10.2 MUIA\_Cycle\_Entries – (V4) [I..], STRPTR \*****FUNCTION**

Here you can define what entries shall be displayed in your cycle gadget. You must supply a pointer to a string array, containing one entry for each item and terminated with a NULL.

Remember that cycle gadget entries may contain any text formatting code such as bold, italic or underlined characters.

Cycle gadgets set the preparse string for all entries to "\33c", this means that they will automatically appear centered. Of course you can override this by simply preceding your entries with own formatting code.

**EXAMPLE**

```
static const char *CYA_GroupTitleColor[] =
{
    "normal",
    "highlight",
    "3-dimensional",
    NULL
};

CY_Title = CycleObject,
MUIA_Cycle_Entries, CYA_GroupTitleColor,
End;
```

**SEE ALSO**

MUIA\_Cycle\_Active, MUIA\_Text\_Contents

**11 Dirlist.mui**

Dirlist class provides a quick and easy way of showing entries in a directory. It features lots of control attributes, many of them known from the popular asl file requester.

This class is *\*not\** intended to replace `asl.library!` Nobody wants to see every MUI application coming with another selfmade file requester. Please continue using ASL for real file requesting purposes!

However, sometimes it may be useful to have a little directory list placed somewhere in your user interface. Imagine an answering machine tool that stores incoming calls in a preconfigured directory. Using a `dirlist` object, you can include the GUI for selecting a call in your window with lots of other gadgets like "Play", "Delete", etc.

`Dirlist` class offers all of a files attributes: name, size, date, time, flags and comment. Using the `MUIA_List.Format` attribute, you can control which of them shall be displayed.

If you want to read the entries of your directory, just send the `dirlist` object a `MUIM_List.GetEntry` method. You will receive a pointer to a struct `FileInfoBlock` which remains valid until your next call to `MUIM_List.GetEntry`.

### 11.1 MUIM\_Dirlist\_ReRead (V4)

#### SYNOPSIS

```
DoMethod(obj,MUIM_Dirlist_ReRead,);
```

#### FUNCTION

Force the `dirlist` object to reread the current directory.

#### EXAMPLE

```
if (NewCallReceived())
    DoMethod(dirlistobj,MUIM_Dirlist_ReRead);
```

#### SEE ALSO

`MUIA_Dirlist_Directory`

### 11.2 MUIA\_Dirlist\_AcceptPattern – (V4) [IS.], STRPTR

#### FUNCTION

Entries not matching this pattern are rejected. Note that the pattern has to be parsed with `dos.library/ParsePatternNoCase()`.

#### SEE ALSO

`MUIA_Dirlist_RejectPattern`, `MUIA_Dirlist_FilterDrawers`

### 11.3 MUIA\_Dirlist\_Directory – (V4) [ISG], STRPTR

#### FUNCTION

Set a new directory for the `dirlist` object. Since reading a directory can take a long long time, MUI delegates this work to a sub task.

Setting this attribute causes the object to clear the current directory (if any) and start loading a new one. `MUIA_Dirlist.Status` will be set to `MUIV_Dirlist.Status_Reading` and the sub task will be launched.

By listening to `MUIA_Dirlist.Status`, you can learn if the directory reading is completed or if something went wrong.

A value of NULL just clears the current directory and sets MUIA\_Dirlist\_Status to MUIV\_Dirlist\_Status\_Invalid.

#### EXAMPLE

```
set(dirobj,MUIA_Dirlist_Directory,"zyxel:incoming");
```

#### SEE ALSO

MUIA\_Dirlist\_Status

### 11.4 MUIA\_Dirlist\_DrawersOnly – (V4) [IS.], BOOL FUNCTION

Indicate whether you only want drawers to be displayed.

#### SEE ALSO

MUIA\_Dirlist\_Directory, MUIA\_Dirlist\_FilesOnly

### 11.5 MUIA\_Dirlist\_FilesOnly – (V4) [IS.], BOOL FUNCTION

Indicate whether you only want files to be displayed.

#### SEE ALSO

MUIA\_Dirlist\_Directory, MUIA\_Dirlist\_DrawersOnly

### 11.6 MUIA\_Dirlist\_FilterDrawers – (V4) [IS.], BOOL FUNCTION

Indicate whether you want drawers matched againsts MUIA\_Dirlist\_RejectPattern and MUIA\_Dirlist\_AcceptPattern.

Defaults to FALSE.

#### SEE ALSO

MUIA\_Dirlist\_RejectPattern, MUIA\_Dirlist\_AcceptPattern

### 11.7 MUIA\_Dirlist\_FilterHook – (V4) [IS.], struct Hook \* FUNCTION

A hook to call for each file encountered. If the function returns TRUE, the file is included in the file list, otherwise it is rejected and not displayed. The function receives the following parameters:

A0	-	(struct Hook *)	-	the hook itself
A1	-	(struct ExAllData *)	-	valid upto ed_Comment
A2	-	(Object *)	-	the dirlist object

All other filter attributes are ignored when a MUIA\_Dirlist\_FilterHook is set.

**SEE ALSO**

MUIA\_Dirlist\_Directory

**11.8 MUIA\_Dirlist\_MultiSelDirs – (V6) [IS.], BOOL****FUNCTION**

Allows multi selection of directories. Defaults to FALSE.

**SEE ALSO**

MUIA\_Dirlist\_FilterDrawers

**11.9 MUIA\_Dirlist\_NumBytes – (V4) [..G], LONG****FUNCTION**

When MUIA\_Dirlist\_Status is MUIV\_Dirlist\_Valid, you can obtain the number of bytes occupied by the directory from this tag.

**SEE ALSO**

MUIA\_Dirlist\_NumFiles, MUIA\_Dirlist\_NumDrawers

**11.10 MUIA\_Dirlist\_NumDrawers – (V4) [..G], LONG****FUNCTION**

When MUIA\_Dirlist\_Status is MUIV\_Dirlist\_Valid, you can obtain the number of drawers in the displayed directory from this tag.

**SEE ALSO**

MUIA\_Dirlist\_NumFiles, MUIA\_Dirlist\_Status

**11.11 MUIA\_Dirlist\_NumFiles – (V4) [..G], LONG****FUNCTION**

When MUIA\_Dirlist\_Status is MUIV\_Dirlist\_Valid, you can obtain the number of files in the displayed directory from this tag.

**SEE ALSO**

MUIA\_Dirlist\_NumDrawers, MUIA\_Dirlist\_Status

**11.12 MUIA\_Dirlist\_Path – (V4) [..G], STRPTR****FUNCTION**

When MUIA\_Dirlist\_Status is MUIV\_Dirlist\_Valid and you have an active entry in the list (MUIA\_List\_Active not equal MUIV\_List\_Active\_Off), you will receive a pointer to the complete path specification of the selected file. Otherwise you get a NULL.

**SEE ALSO**

MUIA\_Dirlist\_Status

### 11.13 MUIA\_Dirlist\_RejectIcons – (V4) [IS.], BOOL FUNCTION

Indicate whether you want icons (\*.info files) to be rejected.

#### SEE ALSO

MUIA\_Dirlist\_Directory

### 11.14 MUIA\_Dirlist\_RejectPattern – (V4) [IS.], STRPTR FUNCTION

Entries matching this pattern are rejected. Note that the pattern has to be parsed with `dos.library/ParsePatternNoCase()`.

#### SEE ALSO

MUIA\_Dirlist\_AcceptPattern, MUIA\_Dirlist\_FilterDrawers

### 11.15 MUIA\_Dirlist\_SortDirs – (V4) [IS.], LONG SPECIAL INPUTS

MUIV\_Dirlist\_SortDirs\_First

MUIV\_Dirlist\_SortDirs\_Last

MUIV\_Dirlist\_SortDirs\_Mix

#### FUNCTION

Adjust the place where directories shall be displayed.

#### SEE ALSO

MUIA\_Dirlist\_SortHighLow, MUIA\_Dirlist\_SortType

### 11.16 MUIA\_Dirlist\_SortHighLow – (V4) [IS.], BOOL FUNCTION

Indicate if you want to sort your directory reversely.

#### SEE ALSO

MUIA\_Dirlist\_SortType, MUIA\_Dirlist\_SortDirs

### 11.17 MUIA\_Dirlist\_SortType – (V4) [IS.], LONG SPECIAL INPUTS

MUIV\_Dirlist\_SortType\_Name

MUIV\_Dirlist\_SortType\_Date

MUIV\_Dirlist\_SortType\_Size

#### FUNCTION

Indicate what fields should be used as sort criteria.

**SEE ALSO**

MUIA\_Dirlist\_SortDirs, MUIA\_Dirlist\_SortHighLow

**11.18 MUIA\_Dirlist\_Status – (V4) [..G], LONG****SPECIAL INPUTS**

MUIV\_Dirlist\_Status\_Invalid  
 MUIV\_Dirlist\_Status\_Reading  
 MUIV\_Dirlist\_Status\_Valid

**FUNCTION**

Read the status of the dirlist object. The result is one of

- MUIV\_Dirlist\_Status\_Invalid: object contains no valid directory.
- MUIV\_Dirlist\_Status\_Reading 1: object is currently reading a new directory.
- MUIV\_Dirlist\_Status\_Valid 2: object contains a valid directory.

**SEE ALSO**

MUIA\_Dirlist\_Directory

**12 Family.mui**

Family class is the base class for objects that are able to handle a list of children. This is e.g. the case for MUIs Menustrip, Menu and MenuItem objects.

Family class defines methods and attributes to add and remove children, sort children, and transfer children to other Family objects.

Group class and application class should also be a subclass of Family class, but due to BOOPSI system limitations, this is currently impossible. If the future will allow more logical class trees, things might change, but everything will be done in a compatible manner.

**12.1 MUIM\_Family\_AddHead (V8)****SYNOPSIS**

DoMethod(obj,MUIM\_Family\_AddHead,Object \*obj);

**FUNCTION**

Add an object as first object to the family. Subclasses of family class usually define which types of objects are possible within their family.

**INPUTS**

**obj** - the object to be added.

**SEE ALSO**

MUIM\_Family\_AddTail, MUIM\_Family\_Insert, MUIM\_Family\_Remove,  
 MUIA\_Family\_Child



## 12.2 MUIM\_Family\_AddTail (V8)

### SYNOPSIS

DoMethod(obj,MUIM\_Family\_AddTail,Object \*obj);

### FUNCTION

Add an object as last object to the family. Subclasses of family class usually define which types of objects are possible within their family.

This method does the same as OM\_ADDMEMBER.

### INPUTS

**obj** - the object to be added.

### SEE ALSO

MUIM\_Family\_AddHead, MUIM\_Family\_Insert, MUIM\_Family\_Remove,  
MUIA\_Family\_Child

## 12.3 MUIM\_Family\_Insert (V8)

### SYNOPSIS

DoMethod(obj,MUIM\_Family\_Insert,Object \*obj, Object \*pred);

### FUNCTION

Add an object after another object to the family. Subclasses of family class usually define which types of objects are possible within their family.

### INPUTS

**obj** - the object to be added.

**pred** - the new object is inserted *after* this object. pred must of course be a member of the family.

### SEE ALSO

MUIM\_Family\_AddTail, MUIM\_Family\_AddHead, MUIM\_Family\_Remove,  
MUIA\_Family\_Child

## 12.4 MUIM\_Family\_Remove (V8)

### SYNOPSIS

DoMethod(obj,MUIM\_Family\_Remove,Object \*obj);

### FUNCTION

Remove an object from a family.

This method does the same as OM\_REMEMBER.

### INPUTS

**obj** - the object to be removed.

**SEE ALSO**

MUIM\_Family\_AddTail, MUIM\_Family\_Insert, MUIM\_Family\_AddHead,  
MUIA\_Family\_Child

**12.5 MUIM\_Family\_Sort (V8)****SYNOPSIS**

DoMethod(obj,MUIM\_Family\_Sort,Object \*obj[1]);

**FUNCTION**

Sort the children of a family.

**INPUTS**

**child** - array that contains \*all\* the children of the family in the desired order.  
The array must be terminated with a NULL entry.

**SEE ALSO**

MUIA\_Family\_Child

**12.6 MUIM\_Family\_Transfer (V8)****SYNOPSIS**

DoMethod(obj,MUIM\_Family\_Transfer,Object \*family);

**FUNCTION**

All the children of the family are removed and added to another family in the same order.

**INPUTS**

**family** - the destination family.

**SEE ALSO**

MUIA\_Family\_Child

**12.7 MUIA\_Family\_Child – (V8) [I.], Object \*****FUNCTION**

You supply a pointer to a previously created MUI object here. This object will be added to family at family creation time.

Of course you can specify any number of child objects, limited only by available memory.

Normally, the value for a MUIA\_Family\_Child tag is a direct call to another MUI\_NewObject(), children are generated "on the fly".

When a family is disposed, all of its children will also get deleted. If you supply a NULL pointer as child, the family object will fail and previously dispose all valid children found in the taglist.

This behaviour makes it possible to generate a complete family within one single (but long) MUI\_NewObject() call. Error checking is not necessary since every error,

even if it occurs in a very deep nesting level, will cause the complete call to fail without leaving back any previously created object.

#### **NOTE**

As a special case, `MUIA_Group_Child` is also recognized and treated as `MUIA_Family_Child`.

#### **SEE ALSO**

`MUIM_Family_AddTail`, `MUIM_Family_Insert`, `MUIM_Family_AddHead`,  
`MUIA_Family_Remove`

## **13 Floattext.mui**

`Floattext` class is a subclass of list class that takes a big text string as input and splits it up into several lines to be displayed. Formatting capabilities include paragraphs and justified text with word wrap.

### **13.1 MUIA\_Floattext\_Justify – (V4) [ISG], BOOL**

#### **FUNCTION**

Indicate whether you want your text aligned to the left and right border. MUI will try to insert spaces between words to reach this goal.

If you want right aligned or centered text, use the `MUIA_List_Format` attribute.

#### **SEE ALSO**

`MUIA_Floattext_Text`, `MUIA_List_Format`

### **13.2 MUIA\_Floattext\_SkipChars – (V4) [IS.], STRPTR**

#### **FUNCTION**

Defines an array of characters that shall be skipped when displaying the text. If you e.g. want to display a fido message and know it has some CTRL-A control characters in it, you could set this attribute to `"\1"` to prevent `floattext` class from displaying unreadable crap.

#### **SEE ALSO**

`MUIA_Floattext_Text`

### **13.3 MUIA\_Floattext\_TabSize – (V4) [IS.], LONG**

#### **FUNCTION**

Adjust the tab size for a text. The tab size is measured in spaces, so if you plan to use tabs not only at the beginning of a paragraph, you should consider using the fixed width font.

Tab size defaults to 8.

#### **SEE ALSO**

`MUIA_Floattext_Text`

### 13.4 MUIA\_Floattext\_Text – (V4) [ISG], STRPTR

#### FUNCTION

String of characters to be displayed as floattext. This string may contain linefeeds to mark the end of paragraphs or tab characters for indention.

MUI will automatically format the text according to the width of the floattext object. If a word won't fit into the current line, it will be wrapped.

If you plan to use tabs not only at the beginning of a line you should consider using the configured fixed width font.

MUI copies the complete string into a private buffer, you won't need to keep your text in memory. If memory is low, nothing will be displayed. That's why you always have to be prepared for handling a NULL pointer when getting back MUIA\_Floattext\_Text.

Setting MUIA\_Floattext\_Text to NULL means to clear the current text.

Please note that justification and word wrap with proportional fonts is a complicated operation and may take a considerable amount of time, especially with long texts on slow machines.

#### EXAMPLE

```
char *text = AllocVec(filesize, MEMF_ANY);

Read(file, text, filesize);

fto = FloattextObject,
    MUIA_Floattext_Text, text,
    End;

FreeVec(text);

/* ... if you need your text later, you can get it */
/* with a simple get(fto, MUIA_Floattext_Text, &text); */
```

#### SEE ALSO

MUIA\_Floattext\_Justify, MUIA\_Floattext\_TabSize, MUIA\_Floattext\_SkipChars

## 14 Gauge.mui

A gauge object is a nice looking display element useful for some kind of progress display.

### 14.1 MUIA\_Gauge\_Current – (V4) [ISG], LONG

#### FUNCTION

Set the current level of the gauge. The value must be between 0 and MUIA\_Gauge\_Max.

#### SEE ALSO

MUIA\_Gauge\_Max

## 14.2 MUIA\_Gauge\_Divide – (V4) [ISG], BOOL

### FUNCTION

If this attribute is != 0, every value set with MUIA\_Gauge\_Current will be divided by this before further processing.

### EXAMPLE

See BoopsiDoor demo program.

### SEE ALSO

MUIA\_Gauge\_Current

## 14.3 MUIA\_Gauge\_Horiz – (V4) [I.], BOOL

### FUNCTION

Determine if you want a horizontal or vertical gauge. Default to FALSE

### SEE ALSO

MUIA\_Gauge\_Current

## 14.4 MUIA\_Gauge\_InfoText – (V7) [ISG], char \*

### FUNCTION

The text given here is displayed within a gauge object and is usually intended to show some kind of percentage information.

This texts preparse is set to "\33c\0338", this makes it appear centered and highlighted by default.

Any %ld will be replaced with the current value of MUIA\_Gauge\_Current.

### NOTE

Currently, InfoText works only for horizontal gauges and gives them a fixed height.

### VERSION

Implemented in version 7 of gauge class (MUI 1.5).

### EXAMPLE

```
...
MUIA_Gauge_InfoText, "%ld %%",
...
```

### SEE ALSO

MUIA\_Gauge\_Current

### 14.5 MUIA\_Gauge\_Max – (V4) [ISG], LONG

#### FUNCTION

Set the maximum value for the gauge. Defaults to 100.

#### SEE ALSO

MUIA\_Gauge\_Current

## 15 Group.mui

Group class is responsible for the complete layout of a MUI window. A group may contain any number of child objects, maybe buttons, cycle gadgets or even other groups.

Some attributes of group class define how the children of a group are layouted. You can e.g. tell your group to place its children horizontally (in a row) or vertically (in a column). Since every MUI object knows about its minimum and maximum dimensions, group class has everything it needs to do that job.

More sophisticated layout is possible by assigning different weights to objects in a group or by making a group two-dimensional.

Beneath the layout issues, a group object passes attributes and methods through to all of its children. Thus, you can talk and listen to any child of a group by talking and listening to the group itself.

### 15.1 MUIA\_Group\_ActivePage – (V5) [ISG], LONG

#### SPECIAL INPUTS

MUIV\_Group\_ActivePage\_First

MUIV\_Group\_ActivePage\_Last

MUIV\_Group\_ActivePage\_Prev

MUIV\_Group\_ActivePage\_Next

#### VERSION

Available since version 5 of "group.mui".

#### FUNCTION

Set (or get) the active page of a page group. Only this active page is displayed, all others are hidden.

The value may range from 0 (for the first child) to numchildren-1 (for the last child). Children are addressed in the order of creation:

#### EXAMPLE

```
PageGroup,
  Child, Page_0_Object,
  Child, Page_1_Object,
  Child, Page_2_Object,
  Child, Page_3_Object,
End;
```

#### NOTE

You may **never** supply an incorrect page value!

**SEE ALSO**

MUIA\_Group\_PageMode

**15.2 MUIA\_Group\_Child – (V4) [I.], Object \*****FUNCTION**

You supply a pointer to a previously created MUI object here. This object will be treated as child of the group, the group is responsible for positioning the object.

Of course you can specify any number of child objects, limited only by available memory.

Normally, the value for a MUIA\_Group\_Child tag is a direct call to another MUI\_NewObject(), children are generated "on the fly".

When a group is disposed, all of its children will also get deleted. If you supply a NULL pointer as child, the group object will fail and previously dispose all valid children found in the taglist.

This behaviour makes it possible to generate a complete application within one single (but long) MUI\_NewObject() call. Error checking is not necessary since every error, even if it occurs in a very deep nesting level, will cause the complete call to fail without leaving back any previously created object.

**EXAMPLE**

Please have a look at some of the supplied example programs.

**SEE ALSO**

MUIA\_Group\_Horiz

**15.3 MUIA\_Group\_Columns – (V4) [IS.], LONG****FUNCTION**

Indicate number of columns in a two dimensional group. If you use this tag, the total number of children must be dividable by the number of columns.

The children will be positioned in a two dimensional array, e.g. allowing easy creation of button fields (maybe for calculator).

The children in your taglist are always read line by line.

When MUI layouts two-dimensional groups, it does actually two layout calculations, one for the rows and one the columns. Parameters like weights and dimensions are handled this way:

- the minimum width of a column/row is the maximum minimum width of all objects in this column/row.
- the maximum width of a column/row is the minimum maximum width of all objects in this column/row.
- the weight of a column/row is the sum of all objects in this column/row.

Actually, there is no difference if you use MUIA\_Group\_Columns or MUIA\_Group\_Rows.

**EXAMPLE**

```

/* group of labeled string gadgets */

GroupObject,
    MUIA_Group_Columns, 2,
    MUIA_Group_Child   , label1,
    MUIA_Group_Child   , string1,
    MUIA_Group_Child   , label2,
    MUIA_Group_Child   , string2,
    MUIA_Group_Child   , label3,
    MUIA_Group_Child   , string3,
    ...
End;

```

**SEE ALSO**

MUIA\_Group\_Rows, MUIA\_Group\_Horiz

**15.4 MUIA\_Group\_Horiz – (V4) [I.], BOOL****FUNCTION**

Boolean value to indicate whether the objects in this group shall be layouted horizontally or vertically. Defaults to FALSE.

This is the easy way of telling your group how it has to look like. If you want two-dimensional groups, you have to use MUIA\_Group\_Columns or MUIA\_Group\_Rows.

**EXAMPLE**

```

GroupObject,
    MUIA_Group_Horiz, TRUE,
    MUIA_Group_Child, obj1,
    MUIA_Group_Child, obj2,
    MUIA_Group_Child, obj3,
End;

```

**SEE ALSO**

MUIA\_Group\_Columns, MUIA\_Group\_Rows, MUIA\_Group\_Child

**15.5 MUIA\_Group\_HorizSpacing – (V4) [IS.], LONG****FUNCTION**

Number of pixels to be inserted between horizontal elements of a group.

Please use this tag wisely, you will override the user's preferred default setting!

**SEE ALSO**

MUIA\_Group\_Spacing, MUIA\_Group\_VertSpacing



## 15.6 MUIA\_Group\_PageMode – (V5) [IS.], BOOL

### VERSION

Available since version 5 of "group.mui".

### FUNCTION

Settings this attribute to TRUE makes the current group a page group. Page groups always display only one their children, which one can be adjusted with the MUIA\_Group\_ActivePage attribute.

Imagine you have a preferences window with several different pages, e.g. the MUI preferences with object, frame, image, font, screen, keyboard and system prefs. Instead of one separate window for each group, you could put all pages into one page group and have a cycle gadget for page switching. This will make your program easier to use since the user won't have to handle a lot of windows. However, he will not be able to work with more than one page at the same time.

Sizes are calculated as follows:

- The minimum width/height of a page group is the maximum minimum width/height of all its children.
- The maximum width/height of a page group is the minimum maximum width/height of all its children.
- When the maximum width/height of a child in a page group is smaller than the minimum width/height of the page group (since it contains another child with big minimum width/height), the child be centered.

Page groups are not limited in depth, children of a page group may of course be other page groups.

If you want to have a gadget only visible under certain conditions, you could make a page group containing this gadget and an empty rectangle object.

If you want TAB cycling for the objects in a page group, simply include all objects in the cycle chain (as if they all were visible at the same time).

### EXAMPLE

demo program "Pages.c"

### SEE ALSO

MUIA\_Group\_ActivePage

## 15.7 MUIA\_Group\_Rows – (V4) [IS.], LONG

### FUNCTION

Indicate number of rows in a two dimensional group. If you use this tag, the total number of children must be dividable by the number of rows.

The children will be positioned in a two dimensional array, e.g. allowing easy creation of button fields (maybe for calculator).

The children in your taglist are always read line by line.

When MUI layouts two-dimensional groups, it does actually two layout calculations, one for the rows and one the columns. Parameters like weights and dimensions are handled this way:

- the minimum width of a column/row is the maximum minimum width of all objects in this column/row.
- the maximum width of a column/row is the minimum maximum width of all objects in this column/row.
- the weight of a column/row is the sum of all objects in this column/row.

Actually, there is no difference if you use `MUIA_Group_Columns` or `MUIA_Group_Rows`.

#### SEE ALSO

`MUIA_Group_Columns`, `MUIA_Group_Horiz`

### 15.8 MUIA\_Group\_SameHeight – (V4) [I.], BOOL

#### FUNCTION

Boolean value to indicate that all children of this group shall have the same height.

#### BUGS

Up to version 5 of groupclass, using `MUIA_Group_SameHeight` could make objects larger than their maximum height. This has been fixed for version 6.

#### SEE ALSO

`MUIA_Group_SameSize`, `MUIA_Group_SameWidth`

### 15.9 MUIA\_Group\_SameSize – (V4) [I.], BOOL

#### FUNCTION

This is a shorthand for `MUIA_Group_SameWidth` and `MUIA_Group_SameHeight`, it sets both of these attributes at once.

Using `MUIA_Group_SameSize`, you won't need to think if your group is horizontal or vertical, both cases are handled automatically.

Forcing all objects of a group to be the same size is e.g. useful for a row of buttons. It's visually more attractive when these buttons have equal sizes instead of being just as big as the text within.

#### BUGS

Up to version 5 of groupclass, using `MUIA_Group_SameSize` could make objects larger than their maximum size. This has been fixed for version 6.

#### EXAMPLE

```
/* three buttons, same size */

GroupObject,
    MUIA_Group_Horiz    , TRUE,
    MUIA_Group_SameSize, TRUE,
    MUIA_Group_Child    , but1,
    MUIA_Group_Child    , but2,
    MUIA_Group_Child    , but3,
```

End;

## SEE ALSO

MUIA\_Group\_SameWidth, MUIA\_Group\_SameHeight

## 15.10 MUIA\_Group\_SameWidth – (V4) [I.], BOOL

### FUNCTION

Boolean value to indicate that all children of this group shall have the same width.

### BUGS

Up to version 5 of groupclass, using MUIA\_Group\_SameWidth could make objects larger than their maximum width. This has been fixed for version 6.

## SEE ALSO

MUIA\_Group\_SameSize, MUIA\_Group\_SameHeight

## 15.11 MUIA\_Group\_Spacing – (V4) [IS.], LONG

### FUNCTION

This is a shorthand for MUIA\_Group\_HorizSpacing and MUIA\_Group\_VertSpacing, it sets both of these attributes at once.

Using MUIA\_Group\_Spacing, you won't need to think if your group is horizontal or vertical, both cases are handled automatically.

Note that setting a spacing value for a group overrides the user's default settings. Please use it only if you have a good reason.

## EXAMPLE

```
/* no space between obj1 and obj2: */
```

```
GroupObject,
  MUIA_Group_Horiz  , TRUE,
  MUIA_Group_Spacing, 0,
  MUIA_Group_Child  , obj1,
  MUIA_Group_Child  , obj2,
End;
```

## SEE ALSO

MUIA\_Group\_HorizSpacing, MUIA\_Group\_VertSpacing

## 15.12 MUIA\_Group\_VertSpacing – (V4) [IS.], LONG

### FUNCTION

Number of pixels to be inserted between vertical elements of a group.

Please use this tag wisely, you will override the user's preferred default setting!

**SEE ALSO**

MUIA\_Group\_Spacing, MUIA\_Group\_HorizSpacing

**16 Image.mui**

Image class is used to display one of MUI's standard images or some selfmade image data.

**16.1 MUIA\_Image\_FontMatch – (V4) [I..], BOOL****FUNCTION**

If TRUE, width and height of the given image will be scaled to match the current font. Images are always defined with a reference font of topaz/8, bigger fonts will make the image grow (as long as its maximum size is big enough).

**EXAMPLE**

The arrows of a scroll bar are e.g. defined with MUIA\_Image\_FontMatch.

**SEE ALSO**

MUIA\_Image\_FontMatch, MUIA\_Image\_FontMatchWidth

**16.2 MUIA\_Image\_FontMatchHeight – (V4) [I..], BOOL****FUNCTION**

If TRUE, the height of the given image will be scaled to match the current font. Images are always defined with a reference font of topaz/8, bigger fonts will make the image grow (as long as its maximum size is big enough).

**SEE ALSO**

MUIA\_Image\_FontMatch, MUIA\_Image\_FontMatchWidth

**16.3 MUIA\_Image\_FontMatchWidth – (V4) [I..], BOOL****FUNCTION**

If TRUE, the width of the given image will be scaled to match the current font. Images are always defined with a reference font of topaz/8, bigger fonts will make the image grow (as long as its maximum size is big enough).

**SEE ALSO**

MUIA\_Image\_FontMatch, MUIA\_Image\_FontMatchHeight

**16.4 MUIA\_Image\_FreeHoriz – (V4) [I..], BOOL****FUNCTION**

Tell the image if its allowed to get scaled horizontally. Defaults to FALSE.

**SEE ALSO**

MUIA\_Image\_FreeVert, MUIA\_Image\_FontMatch

**16.5 MUIA\_Image\_FreeVert – (V4) [I..], BOOL****FUNCTION**

Tell the image if its allowed to get scaled vertically. Defaults to FALSE.

**SEE ALSO**

MUIA\_Image\_FreeHoriz, MUIA\_Image\_FontMatch

**16.6 MUIA\_Image\_OldImage – (V4) [I..], struct Image \*****FUNCTION**

Allows you to use any conventional image structure within a MUI window. The resulting object is always as big as the image and not resizable.

**16.7 MUIA\_Image\_Spec – (V4) [I..], char \*****FUNCTION**

Specify the type of your image. Usually, you will use one of the predefined standard images here, (one of the MUII\_xxx definitions from mui.h), but you also can supply a string containing a MUI image specification. Image specifications always starts with a digit, followed by a ':', followed by some parameters. Currently, the following things are defined (all numeric parameters need to be ascii values!):

**"0:<x>"** where <x> is between MUII\_BACKGROUND and MUII\_FILLBACK2 identifying a builtin pattern.

**"1:<x>"** where <x> identifies a builtin standard image. Don't use this, use **"6:<x>"** instead.

**"2:<r>,<g>,<b>"** where <r>, <g> and <b> are 32-bit RGB color values specified as 8-digit hex string (e.g. 00000000 or ffffffff). Kick 2.x users will get an empty image.

**"3:<n>"** where <n> is the name of an external boopsi image class.

**"4:<n>"** where <n> is the name of an external MUI brush.

**"5:<n>"** where <n> is the name of an external picture file that should be loaded with datatypes. Kick 2.x users will get an empty image.

**"6:<x>"** where <x> is between MUII\_WindowBack and MUII\_Count-1 identifying a preconfigured image/background.

**SEE ALSO**

MUIA\_Image\_OldImage

## 16.8 MUIA\_Image\_State – (V4) [IS.], LONG

### FUNCTION

Some MUI images offer different states, you can select one of the by setting this attribute. Simply use one of the IDS\_NORMAL, IDS\_SELECTED, ... values defined in "intuition/imageclass.h".

### NOTE

Objects that respond to user input will automatically toggle their state between IDS\_NORMAL to IDS\_SELECTED depending on their MUIA\_Selected attribute.

### SEE ALSO

MUIA\_Image\_Spec

## 17 List.mui

MUI's list class is very powerful. It handles all types of entries, from a simple string to a complicated structure with many associated resources. Multi column lists are also supported, the format for a column is adjustable.

Lists support any kind of sorting, multi selection and an active entry that can be controlled with the mouse or the cursor keys.

Note: A list object alone doesn't make much sense, you should always use it as child of a listview object. This one attaches a scrollbar and handles all user input.

### 17.1 MUIM\_List\_Clear (V4)

#### SYNOPSIS

```
DoMethod(obj,MUIM_List_Clear,);
```

#### FUNCTION

Clear the list, all entries are removed. If a destruct hook is set it will be called for every entry.

#### SEE ALSO

MUIM\_List\_Insert, MUIA\_List\_DestructHook

### 17.2 MUIM\_List\_Exchange (V4)

#### SYNOPSIS

```
DoMethod(obj,MUIM_List_Exchange,LONG pos1, LONG pos2);
```

#### FUNCTION

Exchange two entries in a list.

**INPUTS**

**pos1** - number of the first entry.

**pos2** - number of the second entry.

Possible special values since muimaster.library V9:

MUIV_List_Exchange_Top	0	
MUIV_List_Exchange_Active	-1	
MUIV_List_Exchange_Bottom	-2	
MUIV_List_Exchange_Next	-3	/* only valid for second parameter */
MUIV_List_Exchange_Previous	-4	/* only valid for second parameter */

**SEE ALSO**

MUIM\_List\_Insert, MUIM\_List\_Remove, MUIM\_List\_Move

**17.3 MUIM\_List\_GetEntry (V4)****SYNOPSIS**

```
DoMethod(obj,MUIM_List_GetEntry,LONG pos, APTR *entry);
```

**FUNCTION**

Get an entry of a list.

**INPUTS**

**pos** - Number of entry, MUIV\_List\_GetEntry\_Active can be used to get the active entry.

**entry** - Pointer to a longword where the entry will be stored. If the entry is not available (either because you are out of bounds or because there is no active entry), you will receive a NULL.

**EXAMPLE**

```
/* iterate through a list containing file info blocks */

for (i=0;;i++)
{
    struct FileInfoBlock *fib;

    DoMethod(list,MUIM_List_GetEntry,i,&fib);
    if (!fib) break;

    printf("%s\n",fib->fib_FileName);
}
```

**SEE ALSO**

MUIM\_List\_Insert, MUIM\_List\_Remove

## 17.4 MUIM\_List\_Insert (V4)

### SYNOPSIS

DoMethod(obj,MUIM\_List\_Insert,APTR \*entries, LONG count, LONG pos);

### FUNCTION

Insert new entries into a list. When the list has a construct hook, the given pointers won't be inserted directly but instead passed through to the construct hook.

### INPUTS

**entries** - pointer to an array of pointers to be inserted. Warning: This is a pointer to a pointer. See example for details.

**count** - Number of elements to be inserted. If count==-1, entries will be inserted until NULL pointer in the entries array is found.

**pos** - New entries will be added in front of this entry.

MUIV_List_Insert_Top:	insert as first entry.
MUIV_List_Insert_Active:	insert in front of the active entry.
MUIV_List_Insert_Sorted:	insert sorted.
MUIV_List_Insert_Bottom:	insert as last entry.

### EXAMPLE

```
/* insert a string */
char *str = "New entry";
DoMethod(list,MUIM_List_Insert,&str,1,MUIV_List_Insert_Bottom);

/* insert an array */
char *str[] =
{
    "Entry 1",
    "Entry 2",
    "Entry 3",
    "Entry 4",
    NULL
};
DoMethod(list,MUIM_List_Insert,str,-1,MUIV_List_Insert_Bottom);
```

### SEE ALSO

MUIM\_List\_Remove, MUIA\_List\_ConstructHook

## 17.5 MUIM\_List\_InsertSingle (V7)

### SYNOPSIS

DoMethod(obj,MUIM\_List\_InsertSingle,APTR entry, LONG pos);



**FUNCTION**

Insert one new entry into a list. Using MUIM\_List\_Insert has caused some confusion since it takes an array to items instead a single item. To insert single items, MUIM\_List\_InsertSingle is the better choice.

When the list has a construct hook, the given pointer won't be inserted directly but instead passed through to the construct hook.

**INPUTS**

**entry** - item to insert.

**pos** - New entry will be added in front of this entry.

MUIV_List_Insert_Top:	insert as first entry.
MUIV_List_Insert_Active:	insert in front of the active entry.
MUIV_List_Insert_Sorted:	insert sorted.
MUIV_List_Insert_Bottom:	insert as last entry.

**EXAMPLE**

```
/* insert a string */
DoMethod(list,MUIM_List_InsertSingle,"foobar",MUIV_List_Insert_Bottom);
```

**SEE ALSO**

MUIM\_List\_Remove, MUIA\_List\_ConstructHook, MUIM\_List\_InsertSingle

**17.6 MUIM\_List\_Jump (V4)****SYNOPSIS**

```
DoMethod(obj,MUIM_List_Jump,LONG pos);
```

**FUNCTION**

Scroll any entry into the visible part of a list.

**NOTE**

Jumping to an entry doesn't mean to make this entry the active one. This can be done by setting the MUIA\_List\_Active attribute.

**INPUTS**

**pos** - Number of the entry that should be made visible. Use MUIV\_List\_Jump\_Active to jump to the active entry.

**EXAMPLE**

```
/* line 42 is interesting, so make it visible */
DoMethod(list,MUIM_List_Jump,42);
```

**SEE ALSO**

MUIA\_List\_Active

## 17.7 MUIM\_List\_Move (V9)

### SYNOPSIS

DoMethod(obj,MUIM\_List\_Move, LONG from, LONG to);

### FUNCTION

Move an entry from one position to another.

### INPUTS

**pos1** - number of the first entry.

**pos2** - number of the second entry.

Possible special values since muimaster.library V9:

MUIV_List_Move_Top	0	
MUIV_List_Move_Active	-1	
MUIV_List_Move_Bottom	-2	
MUIV_List_Move_Next	-3	/* only valid for second parameter */
MUIV_List_Move_Previous	-4	/* only valid for second parameter */

### SEE ALSO

MUIM\_List\_Insert, MUIM\_List\_Remove, MUIM\_List\_Exchange

## 17.8 MUIM\_List\_NextSelected (V6)

### SYNOPSIS

DoMethod(obj,MUIM\_List\_NextSelected, LONG \*pos);

### FUNCTION

Iterate through the selected entries of a list. This method steps through the contents of a (multi select) list and returns every entry that is currently selected. When no entry is selected but an entry is active, only the active entry will be returned.

This behaviour will result in not returning the active entry when you have some other selected entries somewhere in your list. Since the active entry just acts as some kind of cursor mark, this seems to be the only sensible possibility to handle multi selection together with keyboard control.

### INPUTS

**pos** - a pointer to longword that will hold the number of the returned entry. Must be set to MUIV\_List\_NextSelected\_Start at start of iteration. Is set to MUIV\_List\_NextSelected\_End when iteration is finished.

### EXAMPLE

```
/* Iterate through a list with FileInfoBlocks */

struct FileInfoBlock *fib;
LONG id = MUIV_List_NextSelected_Start;
```

```

for (;;)
{
    DoMethod(list,MUIM_List_NextSelected,&id);
    if (id==MUIV_List_NextSelected_End) break;

    DoMethod(list,MUIM_List_GetEntry,id,&fib);
    printf("selected: %s\n",fib->fib_FileName);
}

```

**SEE ALSO**

MUIM\_List\_Select

**17.9 MUIM\_List\_Redraw (V4)****SYNOPSIS**

DoMethod(obj,MUIM\_List\_Redraw,LONG pos);

**FUNCTION**

If you made some changes to an entry of your list and want these changes to be shown in the display, you will have to call this method.

**INPUTS**

**pos** - Number of the line to redraw. When the line is not currently visible, nothing will happen. Specials:

MUIV_List_Redraw_Active:	redraw the active line (if any),
MUIV_List_Redraw_All:	redraw all lines.

**EXAMPLE**

```

/* do a complete refresh: */
DoMethod(list,MUIM_List_Redraw,MUIV_List_Redraw_All);

```

**17.10 MUIM\_List\_Remove (V4)****SYNOPSIS**

DoMethod(obj,MUIM\_List\_Remove,LONG pos);

**FUNCTION**

Remove an entry from a list.

**INPUTS**

**pos** - number of the entry to be removed or one of  
 MUIV\_List\_Remove\_First,  
 MUIV\_List\_Remove\_Active,  
 MUIV\_List\_Remove\_Last.  
 When the active entry is removed, the following entry will become active.

**EXAMPLE**

```
/* when delete is pressed, remove the active entry */
DoMethod(btDel,MUIM_Notify,MUIA_Pressed,FALSE,
    list,2,MUIM_List_Remove,MUIV_List_Remove_Active);
```

**SEE ALSO**

MUIM\_List\_Insert, MUIA\_List\_DestructHook

**17.11 MUIM\_List\_Select (V4)****SYNOPSIS**

DoMethod(obj,MUIM\_List\_Select,LONG pos, LONG seltype, LONG \*state);

**FUNCTION**

Select/deselect a list entry or ask an entry if its selected.

**INPUTS**

**pos** - Number of the entry or

MUIV_List_Select_Active	for the active entry.
MUIV_List_Select_All	for all entries.

**seltype** - Value:

MUIV_List_Select_Off	unselect entry.
MUIV_List_Select_On	select entry.
MUIV_List_Select_Toggle	toggle entry.
MUIV_List_Select_Ask	just ask about the state.

**state** - Pointer to a longword. If not NULL, this will be filled with the current selection state.

**NOTE**

Since version V9 of muimaster.library: If pos==MUIV\_List\_Select\_All and seltype==MUIV\_List\_Select\_Ask, state will be filled with the total number of selected entries.

**EXAMPLE**

```
/* toggle selection state of active entry */
DoMethod(list,MUIM_List_Select,MUIV_List_Select_Active,
    MUIV_List_Select_Toggle,NULL);

/* select all entries */
DoMethod(list,MUIM_List_Select,MUIV_List_Select_All,
    MUIV_List_Select_On,NULL);
```

**SEE ALSO**

MUIA\_List\_MultiTest\_Hook

**17.12 MUIM\_List\_Sort (V4)****SYNOPSIS**

DoMethod(obj,MUIM\_List\_Sort,);

**FUNCTION**

Sort the list. MUI uses an iterative quicksort algorithm, no stack problems will occur.

**SEE ALSO**

MUIA\_List\_CompareHook

**17.13 MUIA\_List\_Active – (V4) [ISG], LONG****SPECIAL INPUTS**

MUIV\_List\_Active\_Off  
 MUIV\_List\_Active\_Top  
 MUIV\_List\_Active\_Bottom  
 MUIV\_List\_Active\_Up  
 MUIV\_List\_Active\_Down  
 MUIV\_List\_Active\_PageUp  
 MUIV\_List\_Active\_PageDown

**FUNCTION**

Reading this attribute will return the number of the active entry (the one with the cursor on it). The result is between 0 and MUIA\_List\_Entries-1 or MUIV\_List\_Active\_Off, in which case there is currently no active entry.

Setting the attribute will cause the list to move the cursor to the new position and scroll this position into the visible area.

**SEE ALSO**

MUIA\_List\_Entries, MUIA\_List\_First, MUIA\_List\_Visible

**17.14 MUIA\_List\_AdjustHeight – (V4) [I.], BOOL****FUNCTION**

A list with MUIA\_List\_AdjustHeight set to true is exactly as high as all of its entries and not resizable. This is only possible when the list is filled *\*before\** the window is opened.

**SEE ALSO**

MUIA\_List\_AdjustWidth

**17.15 MUIA\_List\_AdjustWidth – (V4) [I.], BOOL****FUNCTION**

A list with MUIA\_List\_AdjustWidth set to true is exactly as wide as the widest entry and not resizable. This is only possible when the list is filled **before** the window is opened.

**SEE ALSO**

MUIA\_List\_AdjustHeight

### 17.16 MUIA\_List\_CompareHook – (V4) [IS.], struct Hook \* FUNCTION

If you plan to have the entries of your list sorted (either by inserting them sorted or by using the MUIM\_List\_Sort method) and if the entries of your list are not simple strings, you **must** supply a compare hook.

This hook will be called with one list element in A1 and another one in A2. You should return

return value	statement
-1	e1 < e2
0	e1 == e2
1	e1 > e2

**EXAMPLE**

```
/* the builtin string compare function */

LONG __asm cmpfunc(_a1 char *s1,_a2 char *s2)
{
    return(stricmp(s1,s2));
}
```

**SEE ALSO**

MUIA\_List\_ConstructHook, MUIA\_List\_DestructHook

### 17.17 MUIA\_List\_ConstructHook – (V4) [IS.], struct Hook \*

**SPECIAL INPUTS**

MUIV\_List\_ConstructHook\_String

**FUNCTION**

The construct hook is called whenever you add an entry to your list. MUI will not insert the given pointer directly, but instead call the construct hook and add its result code.

Imagine you want to display a list of entries in a directory. You could step through it using Examine()/ExNext() and directly use the MUIM\_List\_Insert method on your file info block buffer.

Your construct hook will be called with this file info block as parameter, makes a copy of it and returns the address of that copy. That's what is actually added to the list.

The corresponding destruct hook is called whenever an entry shall be removed. Its task would simply be to free the memory and maybe other resources concerning this entry that were allocated by the construct hook.

Using these two functions, you will never have to worry about freeing the memory used by your list entries. Clearing the list or disposing the list object will automatically remove all entries and thus free the associated resources.

The construct hook will be called with the hook in A0, the data given to MUIA\_List\_Insert as message in register A1 and with pointer to a standard kick 3.x memory pool in A2. If you want, you can use the `exec` or `amiga.lib` functions for allocating memory within this pool, but this is only an option.

If the construct hook returns `NULL`, nothing will be added to the list.

There is a builtin construct hook available called `MUIV_List_ConstructHook_String`. This expects that you only add strings to your list and will make a local copy of this string to allow you destroying the original. Of course you **must** also use `MUIV_List_DestructHook_String` in this case.

Without construct and destruct hooks, you are responsible for allocating and freeing entries yourself.

### EXAMPLE

```
/* the builtin string construct and destruct functions: */

APTR __asm consfunc(_a2 APTR pool,_a1 char *str)
{
    char *new;
    if (new=AllocPooled(pool,strlen(str)+1))
        strcpy(new,str);
    return(new);
}

VOID __asm desfunc(_a2 APTR pool,_a1 char *entry)
{
    FreePooled(pool,entry,strlen(entry)+1);
}

/* for more sophisticated hooks see demo program WbMan.c */
```

### SEE ALSO

MUIA\_List\_DestructHook, MUIA\_List\_DisplayHook

## 17.18 MUIA\_List\_DestructHook – (V4) [IS.], struct Hook \*

SPECIAL INPUTS MUIV\_List\_DestructHook\_String

### FUNCTION

Set up a destruct hook for your list. For detailed explanation see `MUIA_List_ConstructHook`.

### SEE ALSO

MUIA\_List\_ConstructHook, MUIA\_List\_DisplayHook

## 17.19 MUIA\_List\_DisplayHook – (V4) [IS.], struct Hook \*

### FUNCTION

Since MUI's lists can handle any kind of entries, you have to supply a display hook to specify what should actually be shown in the display.

The hook will be called with a pointer to the entry to be displayed in A1 and a pointer to a string array containing as many entries as your list may have columns in A2.

You must fill this array with the strings that you want to display.

Note: You can of course use MUI's text engine facilities here to create e.g. right aligned or centered columns.

Without a display hook, MUI expects a simple one columned string list.

See MUIA\_List\_Format for details about column handling.

## NOTE

Since version 6 of MUI, the display hook also gets the position of the current entry as additional parameter. You can easily do e.g. some line numbering using this feature. The number (from 0 to NumEntries-1) is stored in the longword **preceding** the column array (see example below).

## EXAMPLE

```
/* list of file info blocks, two columned, name and size */

LONG __asm dispfunc(_a2 char **array,_a1 struct FileInfoBlock *fib)
{
    static char buf1[20],buf2[20];

    if (fib->fib_EntryType<0)
        sprintf(buf2,"\33r%ld",fib->fib_Size);
    else
        strcpy(buf2,"\33r(dir)");

    sprintf(buf1,"%ld",array[-1]);    // get the line number.

    *array++ = buf1;
    *array++ = fib->fib_FileName;
    *array   = buf2;

    return(0);
}
```

## SEE ALSO

MUIA\_List\_Format, MUIA\_Text\_Contents

## 17.20 MUIA\_List\_Entries – (V4) [..G], LONG FUNCTION

Get the current number of entries in the list.

## SEE ALSO

MUIA\_List\_First, MUIA\_List\_Visible, MUIA\_List\_Active



**17.21 MUIA\_List\_First – (V4) [..G], LONG****FUNCTION**

Get the number of the entry displayed on top of the list. You have to be prepared to get a result of -1, which means that the list is not visible at all (e.g. when the window is iconified).

**SEE ALSO**

MUIA\_List\_Visible, MUIA\_List\_Entries, MUIA\_List\_Active

**17.22 MUIA\_List\_Format – (V4) [ISG], STRPTR****FUNCTION**

MUI has the ability to handle multi column lists. To define how many columns should be displayed and how they should be formatted, you specify a format string.

This format string must contain one entry for each column you want to see. Entries are separated by commas, one entry is parsed via `dos.library/ReadArgs()`.

The template for a single entry looks like this:

DELTA=D/N, PREPARSE=P/K, WEIGHT=W/N, MINWIDTH=MIW/N, MAXWIDTH=MAW/N, COL=C/N

**DELTA** Space in pixel between this column and the next. the last displayed column ignores this setting. Defaults to 4.

**PREPARSE** A preparse value for this column. Setting this e.g. to `"\33c"` would make the column centered. See `MUIA_Text_Contents` for other control codes.

**WEIGHT** The weight of the column. As with MUI's group class, columns are layouted with a minimum size, a maximum size and weight. A column with a weight of 200 would gain twice the space than a column with a weight of 100. Defaults to 100.

**MINWIDTH** Minimum percentage width for the current column. If your list is 200 pixel wide and you set this to 25, your column will at least be 50 pixel. The special value -1 for this parameter means that the minimum width is as wide as the widest entry in this column. This ensures that every entry will be completely visible (as long as the list is wide enough). Defaults to -1.

**MAXWIDTH** Maximum percentage width for the current column. If your list is 200 pixel wide and you set this to 25, your column will not be wider as 50 pixel. The special value -1 for this parameter means that the maximum width is as wide as the widest entry in this column. Defaults to -1.

**COL** This value adjusts the number of the current column. This allows you to adjust the order of your columns without having to change your display hook. See example for details. Defaults to current entry number (0,1,...)

If your list object gets so small there is not enough place for the minwidth of a column, this column will be hidden completely and the remaining space is distributed between the remaining columns. This is not true if the column is the first column, in this case the entries will simply be clipped.

**NOTE**

You will have as many columns in your list as entries in the format string (i.e. number of commas + 1). Empty entries, e.g. with a format string of ",,,," are perfectly ok.

The default list format is an empty string (""), this means a one column list without special formatting.

**BUGS**

Currently there is a maximum of 64 columns for a list.

**EXAMPLE**

```
/* Three column list without further formatting: */
MUIA_List_Format: ",,,"

/* Three column list, middle column centered: */
MUIA_List_Format: ",P=\33c,"

/* Three column list, display order 2 1 0: */
MUIA_List_Format: "COL=2,COL=1,COL=0"

/* now something more complex. */
/* the display hook defines six entries: */
dispfunc(_a2 char **array,_a1 struct Article *at)
{
    *array++ = at->FromName; // col 0
    *array++ = at->FromPath; // col 1
    *array++ = at->ToName;    // col 2
    *array++ = at->ToPath;    // col 3
    *array++ = at->Date;      // col 4
    *array    = at->Subject;   // col 5
}

/* but we only want to have fromname, date and subject
/* actually displayed, subject should be centered: */
MUIA_List_Format, "COL=0,COL=4,COL=5 P=\33c"

/* maybe this looks kind of silly, why not make our */
/* display hook only fill in these three columns. */
/* well, if you would e.g. make the format string */
/* user configurable and document what your display */
/* hook puts into the array, the user could decide */
/* what columns he actually wants to see. */
/* The supplied example DFView does something like */
/* that. */

/* two column list:      ! Eye      1234 !
                        ! Foot      22 !
                        ! Nose  22331 ! */

MUIA_List_Format, "MAW=100,P=\33r"
```

**SEE ALSO**

MUIA\_List\_DisplayHook, MUIA\_Text\_Contents

**17.23 MUIA\_List\_InsertPosition – (V9) [..G], LONG****FUNCTION**

After insertion of an element with MUIM\_List\_Insert, you can query the position of the new entry by getting this attribute.

**17.24 MUIA\_List\_MultiTestHook – (V4) [IS.], struct Hook \*****FUNCTION**

If you plan to have a multi selecting list but not all of your entries are actually multi selectable (e.g. in a file requester), you can supply a MUIA\_List\_MultiTestHook.

It will be called with a pointer to an entry in A1 and should return TRUE if the entry is multi selectable, FALSE otherwise.

**EXAMPLE**

```
/* multi test func for a list of file info blocks */

LONG __asm mtfunc(_a1 struct FileInfoBlock *fib)
{
    if (fib->fib_DirEntryType<0)
        return(TRUE);
    else
        return(FALSE);
}
```

**SEE ALSO**

MUIA\_List\_ConstructHook, MUIA\_List\_DestructHook

**17.25 MUIA\_List\_Quiet – (V4) [.S.], BOOL****FUNCTION**

If you add/remove lots of entries to/from a currently visible list, this will cause lots of screen action and slow down the operation. Setting MUIA\_List\_Quiet to true will temporarily prevent the list from being refreshed, this refresh will take place only once when you set it back to false again.

**EXAMPLE**

```
set(list,MUIA_List_Quiet,TRUE);
AddThousandEntries(list);
set(list,MUIA_List_Quiet,FALSE);
```

**SEE ALSO**

MUIM\_List\_Insert, MUIM\_List\_Remove

**17.26 MUIA\_List\_SourceArray – (V4) [I.], APTR****FUNCTION**

The NULL terminated array given here is immediately inserted into the list after object creation time.

**EXAMPLE**

```
static const char *KeyList[] =
{
    "Cursor Up",
    "Cursor Down",
    "Cursor Left",
    "Cursor Right",
    NULL;
};

LV_Keys = ListviewObject,
    MUIA_Listview_List, ListObject,
    InputListFrame,
    MUIA_List_AdjustWidth, TRUE,
    MUIA_List_SourceArray, KeyList,
    End,
End;
```

**17.27 MUIA\_List\_Title – (V6) [ISG], char \*****FUNCTION**

Specify a title for the current list. The title is displayed at the very first line and doesn't scroll away when the list top position moves.

Usually, the title is just a string. However, if you have a multi column list with a custom display hook and you want to have separate titles for each of your columns, you can set this attribute to TRUE. In this case, whenever MUI feels that the list title has to be drawn, it will call your display hook with a NULL entry pointer. Your hook has to check for this NULL entry and fill the given string array with your column titles. Layout of the column titles follows the same rules as layout of the lists entries.

**EXAMPLE**

```
/* display function for a multi columned file list with titles */

LONG __asm DisplayFunc(_a2 char **array, _a1 struct Entry *e)
{
    struct Data *data = hook->h_Data;

    if (e)
    {
        *array++ = e->Name;
        *array++ = e->Size;
        *array++ = e->Date;
        *array++ = e->Time;
        *array++ = e->Flags;
        *array  = e->Comment;
    }
    else
    {
        *array++ = "Name";
        *array++ = "Size";
        *array++ = "Date";
        *array++ = "Time";
    }
}
```

```

        *array++ = "Flags";
        *array    = "Comment";
    }

    return(0);
}

```

**SEE ALSO**

MUIA\_List\_DisplayHook

**17.28 MUIA\_List\_Visible – (V4) [..G], LONG****FUNCTION**

Get the current number of visible entries in the list. You have to be prepared to get a result of -1, which means that the list is not visible at all (e.g. when the window is iconified).

**SEE ALSO**

MUIA\_List\_First, MUIA\_List\_Entries, MUIA\_List\_Active

**18 Listview.mui**

It's important to know that MUI makes a difference between a list and a listview. A list is just a collection of some entries and is part of a listview, which attaches a scrollbar and input handling to the list.

During object creation time, you have to be careful not specifying listview tags for the list object or list tags for the listview object, both versions won't work. Once the objects are setup, you can of course talk to the listview as if it was the list directly.

**18.1 MUIA\_Listview\_ClickColumn – (V7) [..G], LONG****FUNCTION**

When using a multi column list, this attribute contains the number of the column where the user clicked.

**SEE ALSO**

MUIA\_Listview\_DefClickColumn

**18.2 MUIA\_Listview\_DefClickColumn – (V7) [ISG], LONG****FUNCTION**

When the listview is controlled with the keyboard and the user presses RETURN, the value given here will be used as default for MUIA\_Listview\_ClickColumn.

**SEE ALSO**

MUIA\_Listview\_ClickColumn

### 18.3 MUIA\_Listview\_DoubleClick – (V4) [I.G], BOOL

#### FUNCTION

This attribute is set to TRUE whenever the user double clicks on an entry in the list.

#### SEE ALSO

MUIA\_Listview\_SelectChange

### 18.4 MUIA\_Listview\_Input – (V4) [I.], BOOL

#### FUNCTION

Setting this to FALSE will result in a read only list view. Defaults to TRUE.

#### SEE ALSO

MUIA\_Listview\_MultiSelect

### 18.5 MUIA\_Listview\_List – (V4) [I.], Object \*

#### FUNCTION

Every listview needs a list object as child. Specify it here.

As every other child, it will get disposed when its parent object is disposed.

#### EXAMPLE

```
ListviewObject,
  MUIA_Listview_Input, FALSE,
  MUIA_Listview_List , ListObject,
    ReadListFrame,
    MUIA_List_Format      , ",, ",
  End,
End;
```

#### SEE ALSO

MUIA\_Listview\_Input

### 18.6 MUIA\_Listview\_MultiSelect – (V7) [I.], LONG

#### SPECIAL INPUTS

MUIV\_Listview\_MultiSelect\_None  
 MUIV\_Listview\_MultiSelect\_Default  
 MUIV\_Listview\_MultiSelect\_Shifted  
 MUIV\_Listview\_MultiSelect\_Always

#### FUNCTION

Four possibilities exist for a listviews multi select capabilities:

**MUIV\_Listview\_MultiSelect\_None:** The listview cannot multiselect at all.

**MUIV\_Listview\_MultiSelect\_Default:** The multi select type (with or without shift) depends on the users preferences setting.

**MUIV\_Listview\_MultiSelect\_Shifted:** Overrides the users prefs, multi selecting only together with shift key.

**MUIV\_Listview\_MultiSelect\_Always:** Overrides the users prefs, multi selecting without shift key.

Please do **not** override the users prefs unless you have a good reason!

#### SEE ALSO

MUIA\_List\_MultiTestHook

### 18.7 MUIA\_Listview\_ScrollerPos – (V10) [I..], BOOL

#### SPECIAL INPUTS

MUIV\_Listview\_ScrollerPos\_Default

MUIV\_Listview\_ScrollerPos\_Left

MUIV\_Listview\_ScrollerPos\_Right

#### FUNCTION

Specifies the position of a listviews scrollbar. Don't use this tag unless it's absolutely required!

### 18.8 MUIA\_Listview\_SelectChange – (V4) [..G], BOOL

#### FUNCTION

This attribute is set to TRUE whenever the selection state of one or more items in the list is changing. You can use this e.g. if you want to display the number of selected items in a status line.

#### SEE ALSO

MUIA\_List\_MultiSelect

## 19 Menustrip.mui

Menustrip class is the base class for MUI's object oriented menus. Its children are objects of Menu class, each of them describes exactly one menu.

A Menustrip object doesn't feature many options itself, but as a subclass of Family class, it simply acts as father for multiple Menu objects.

The Menustrip object is usually specified as a child of either Application class or window class with the attributes MUIA\_Application\_Menustrip or MUIA\_Window\_Menustrip.

### 19.1 MUIA\_Menustrip\_Enabled – (V8) [ISG], BOOL

#### FUNCTION

Enable or disable the complete menu strip.

## 20 Menu.mui

Objects of menu class describe exactly one pulldown menu. They don't feature many options themselves, but as a subclass of Family class, they act as father for their several menu item objects.

### 20.1 MUIA\_Menu\_Enabled – (V8) [ISG], BOOL

#### FUNCTION

Enable or disable the complete menu.

### 20.2 MUIA\_Menu\_Title – (V8) [ISG], STRPTR

#### FUNCTION

Describe the title of the menu. Note that the string is **not** copied and must remain valid until the menu object is disposed.

#### SEE ALSO

MUIA\_Menu\_Enabled

## 21 Menuitem.mui

Menuitem class describes a single menu item. You can use all of the gadtools menus features expect Image menus here.

Since Menuitem class is a subclass of Family class, you can add other menu items as children of a menu item to indicate sub menus. MUI does not limit the level of sub menus, but the operating system currently allows a maximum nesting level of one. Because of this, children of menu items should not contain other menu items for now, the results are unpredictable.

#### NOTE

For handling menu items, MUIA\_UserData and the methods MUIM\_SetUserData, MUIM\_GetUserData and MUIM\_FindUserData can become quite useful. See the Menu demo program and the accompanying documentation for details.

### 21.1 MUIA\_Menuitem\_Checked – (V8) [ISG], BOOL

#### FUNCTION

set/get the checked state of a checkit menu item.

#### SEE ALSO

MUIA\_Menuitem\_Checkit, MUIA\_Menuitem\_Enabled, MUIA\_Menuitem\_Exclude

### 21.2 MUIA\_Menuitem\_Checkit – (V8) [ISG], BOOL

#### FUNCTION

Set to TRUE and this item will become a checkmarkable item.



**SEE ALSO**

MUIA\_Menuitem\_Checked, MUIA\_Menuitem\_Enabled, MUIA\_Menuitem\_Exclude

**21.3 MUIA\_Menuitem\_Enabled – (V8) [ISG], BOOL****FUNCTION**

enabled/disalbe the menu item.

**SEE ALSO**

MUIA\_Menuitem\_Checkit, MUIA\_Menuitem\_Checked, MUIA\_Menuitem\_Exclude

**21.4 MUIA\_Menuitem\_Exclude – (V8) [ISG], LONG****FUNCTION**

bitmask of menu item numbers that are to be deselected when this one is selected.

**SEE ALSO**

MUIA\_Menuitem\_Checkit, MUIA\_Menuitem\_Enabled, MUIA\_Menuitem\_Checked

**21.5 MUIA\_Menuitem\_Shortcut – (V8) [ISG], char****FUNCTION**

Define the shortcut for a menu item.

**SEE ALSO**

MUIA\_Menuitem\_Title

**21.6 MUIA\_Menuitem\_Title – (V8) [ISG], STRPTR****FUNCTION**

Define the items title.

**SEE ALSO**

MUIA\_Menuitem\_Shortcut

**21.7 MUIA\_Menuitem\_Toggle – (V8) [ISG], BOOL****FUNCTION**

Define the state of the TOGGLE flag for this item.

**SEE ALSO**

MUIA\_Menuitem\_Checkit, MUIA\_Menuitem\_Enabled, MUIA\_Menuitem\_Checked

## 21.8 MUIA\_Menuitem\_Trigger – (V8) [..G], struct MenuItem \*

### FUNCTION

This attribute is set to a pointer to the struct MenuItem of the item object when the item is selected. By setting up notification on this attribute with MUIV\_EveryTime, you can react on menu actions and query the MenuItems flags immediately.

Note that menu reactions are also possible any maybe a bit easier with MUIA\_Application\_ReturnID, MUIA\_Application\_MenuAction and MUIA\_Window\_MenuAction.

## 22 Notify.mui

Notify class is superclass of all other MUI classes. It's main purpose is to handle MUI's notification mechanism, but it also contains some other methods and attributes useful for every object.

### 22.1 MUIM\_CallHook (V4)

#### SYNOPSIS

```
DoMethod(obj,MUIM_CallHook,struct Hook *Hook, ULONG param1, /* ... */);
```

#### FUNCTION

Call a standard amiga callback hook, defined by a Hook structure. Together with MUIM\_Notify, you can easily bind hooks to buttons, your hook will be called when the button is pressed.

The hook will be called with a pointer to the hook structure in a0, a pointer to the calling object in a2 and a pointer to the first parameter in a1.

#### INPUTS

**Hook** pointer to a struct Hook.

**param1,...** zero or more parameters. The hook function will receive a pointer to the first parameter in register a1.

#### EXAMPLE

standalone:

```
DoMethod(obj,MUIM_CallHook,&hookstruct,13,42,"foobar","barfoo");
```

within a notification statement:

```
DoMethod(propobj,MUIM_Notify,MUIA_Prop_First,MUIV_EveryTime,
        propobj,3,MUIM_CallHook,&prophook,MUIV_TriggerValue);
```

prophook will be called every time the knob is moving and gets a pointer to the knobs current level in a1.

## 22.2 MUIM\_FindUData (V8)

### SYNOPSIS

```
DoMethod(obj,MUIM_FindUData,ULONG udata);
```

### FUNCTION

This method tests if the MUIA\_UserData of the object contains the given <udata> and returns the object pointer in this case.

Although this is not very useful for single objects, performing this method on objects that handle children can become very handy. In this case, all the children (any maybe their children) are tested against <udata> and the first matching object is returned.

This method is especially useful if you created your menu tree with a NewMenu structure and you want to find the object pointer for a single menu item.

### INPUTS

**udata** - userdata to look for.

### RESULT

A pointer to the first object with the specified user data or NULL if no object is found.

### NOTE

If you have many objects in your application, MUIM\_FindUData may take quite long. You can limit the amount of time by performing the method not on the application but on the window or even on the group/family your object is placed in.

### SEE ALSO

MUIM\_GetUData, MUIM\_SetUData

## 22.3 MUIM\_GetUData (V8)

### SYNOPSIS

```
DoMethod(obj,MUIM_GetUData,ULONG udata, ULONG attr, ULONG *storage);
```

### FUNCTION

This method tests if the MUIA\_UserData of the object contains the given <udata> and gets <attr> to <storage> for itself in this case.

Although this is not very useful for single objects, performing this method on objects that handle children can become very handy. In this case, all the children (any maybe their children) are searched against <udata> and the first matching objects will be asked for the specified attribute.

**INPUTS**

**udata** - userdata to look for.

**attr** - attribute to get.

**storage** - place to store the attribute.

**NOTE**

If you have many objects in your application, MUIM\_GetUDData may take quite long. You can limit the amount of time by performing the method not on the application but on the window or even on the group/family your objects are place in.

**SEE ALSO**

MUIM\_SetUDData, MUIM\_FindUDData

**22.4 MUIM\_KillNotify (V4)****SYNOPSIS**

```
DoMethod(obj,MUIM_KillNotify,ULONG TrigAttr);
```

**FUNCTION**

MUIM\_KillNotify kills previously given notifications on specific attributes.

**INPUTS**

**TrigAttr** - Attribute for which the notify was specified. If you set up more than one notify for an attribute, only the first one will be killed.

**EXAMPLE**

```
DoMethod(button,MUIM_KillNotify,MUIA_Pressed);
```

**SEE ALSO**

MUIM\_Notify

**22.5 MUIM\_MultiSet (V7)****SYNOPSIS**

```
DoMethod(obj,MUIM_MultiSet,ULONG attr, ULONG val, APTR obj, /* ... */);
```

**FUNCTION**

Set an attribute for multiple objects. Receiving an attribute/value pair and a list of objects, this method sets the new value for all the objects in the list. This is especially useful for disabling/enabling lots of objects with one singe function call.

The object that executes this method isn't affected!

**NOTE**

This method was implemented in version 7 of notify class.

**INPUTS**

**attr** attribute to set.

**value** new value for the attribute.

**obj, ...** list of MUI objects, terminated with a NULL pointer.

**EXAMPLE**

```
/* disable all the address related gadgets... */

DoMethod(xxx, MUIM_MultiSet, MUIA_Disabled, TRUE,
         ST_Name, ST_Street, ST_City, ST_Country, ST_Phone, NULL);

/* note that the xxx object doesn't get disabled! */
```

**SEE ALSO**

MUIM\_Set, MUIM\_Notify

**22.6 MUIM\_NoNotifySet (V9)****SYNOPSIS**

```
DoMethod(obj, MUIM_NoNotifySet, ULONG attr, char *format, ULONG val, /* ... */);
```

**FUNCTION**

Acts like MUIM\_Set but doesn't trigger any notification. This can become useful to avoid deadlocks with bi-directional connections.

**INPUTS**

**attr** attribute you want to set.

**val** value to set the attribute to.

**EXAMPLE**

```
DoMethod(editor, MUIM_Notify, EDIT_Top, MUIV_EveryTime,
         sbar, 3, MUIM_NoNotifySet, MUIA_Prop_First, MUIV_TriggerValue);

DoMethod(sbar, MUIM_Notify, MUIA_Prop_First, MUIV_EveryTime,
         editor, 3, MUIM_NoNotifySet, EDIT_Top, MUIV_TriggerValue);
```

**SEE ALSO**

MUIM\_Set

**22.7 MUIM\_Notify (V4)****SYNOPSIS**

```
DoMethod(obj, MUIM_Notify, ULONG TrigAttr, ULONG TrigVal, APTR DestObj,
         ULONG FollowParams, /* ... */);
```

## FUNCTION

Add a notification event handler to an object. Notification is essential for every MUI application.

A notification statement consists of a source object, an attribute/value pair, a destination object and a notification method. The attribute/value pair belongs to the source object and determines when the notification method will be executed on the destination object.

Whenever the source object gets the given attribute set to the given value (this can happen because of the user pressing some gadgets or because of your program explicitly setting the attribute with `SetAttrs()`), the destination object will execute the notification method.

With some special values, you can trigger the notification every time the attribute is changing. In this case, you can include the triggering attributes value within the notification method. See below.

One big problem with notification are endless loops. Imagine you have a prop gadget and want to show its state with a gauge object. You connect `MUIA_Prop_First` with `MUIA_Gauge_Max` and everything is fine, the gauge gets updated when the user drags around the gadget. On the other hand, if your program sets the gauge to a new value, you might want your prop gadget to immediately show this change and connect `MUIA_Gauge_Max` with `MUIA_Prop_First`. Voila, a perfect endless loop.

To avoid these conditions, MUI always checks new attribute values against the current state and cancels notification when both values are equal. Thus, setting `MUIA_Prop_First` to 42 if the prop gadgets first position is already 42 won't trigger any notification event.

## INPUTS

**TrigAttr** attribute that triggers the notification.

**TrigValue** value that triggers the notification. The special value `MUIV_EveryTime` makes MUI execute the notification method every time when `TrigAttr` changes. In this case, the special value `MUIV_TriggerValue` in the notification method will be replaced with the value that `TrigAttr` has been set to. You can use `MUIV_TriggerValue` up to four times in one notification method. Since version 8 of `muimaster.library`, you can also use `MUIV_NotTriggerValue` here. In this case, MUI will replace `TRUE` values with `FALSE` and `FALSE` values with `TRUE`. This can become quite useful when you try to set "negative" attributes like `MUIA_Disabled`.

**DestObj** object on which to perform the notification method. Either supply a valid object pointer or one of the following special values (V10) which will be resolved at the time the event occurs:

**MUIV\_Notify\_Self** - notifies the object itself.

**MUIV\_Notify\_Window** - notifies the object's parent window.

**MUIV\_Notify\_Application** - notifies the object's application.

**FollowParams** number of following parameters. If you e.g. have a notification method with three parts (maybe `MUIM_Set,attr,val`), you have to set `FollowParams` to 3. This allows MUI to copy the complete notification method into a private buffer for later use.

... following is the notification method.

**EXAMPLE**

```

/*
** Every time when the user releases a button
** (and the mouse is still over it), the button object
** gets its MUIA_Pressed attribute set to FALSE.
** That's what a program can react on with notification,
** e.g. by opening another window.
*/

DoMethod(buttonobj,MUIM_Notify,
    MUIA_Pressed, FALSE,          /* attribute/value pair */
    windowobj,                    /* destination object */
    3,                            /* 3 following words */
    MUIM_Set, MUIA_Window_Open, TRUE); /* notification method */

/*
** Lets say we want to show the current value of a
** prop gadget somewhere in a text field:
*/

DoMethod(propobj,MUIM_Notify,      /* notification is triggered */
    MUIA_Prop_First, MUIV_EveryTime /* every time the attr changes */
    textobj                    /* destination object */
    4,                          /* 4 following words */
    MUIM_SetAsString, MUIA_Text_Contents,
    "value is %ld !", MUIV_TriggerValue);
/* MUIV_TriggerValue will be replaced with the
   current value of MUIA_Prop_First */

/*
** Inform our application when the user hits return
** in a string gadget:
*/

DoMethod(stringobj,MUIM_Notify,
    MUIA_String_Acknowledge, MUIV_EveryTime,
    MUIV_Notify_Application, 2, MUIM_Application_ReturnID, ID_FOOBAR);

```

**22.8 MUIM\_Set (V4)****SYNOPSIS**

```
DoMethod(obj,MUIM_Set,ULONG attr, ULONG val);
```

**FUNCTION**

Set an attribute to a value. Normally, you would set attributes with `intuition.library SetAttrs()` or with the `OM.SET` method as with any other boopsi objects. But since these calls need a complete tag list, not just a single attribute/value pair, they are not useful within a `MUIM_Notify` method.

**INPUTS**

**attr** attribute you want to set.

**val** value to set the attribute to.

**EXAMPLE**

```
DoMethod(stroobj,MUIM_Set,MUIA_String_Contents,"foobar");
```

and

```
SetAttrs(stroobj,MUIA_String_Contents,"foobar",TAG_DONE);
```

are equal.

**SEE ALSO**

MUIM\_SetAsString, MUIM\_Notify, MUIM\_NoNotifySet

**22.9 MUIM\_SetAsString (V4)****SYNOPSIS**

```
DoMethod(obj,MUIM_SetAsString,ULONG attr, char *format, ULONG val, /* ... */);
```

**FUNCTION**

Set a (text kind) attribute to a string. This can be useful if you want to connect a numeric attribute of an object with a text attribute of another object.

**INPUTS**

**attr** attribute to set.

**format** C like formatting string, remember to use "%ld" !

**val, ...** one or more paremeters for the format string.

**EXAMPLE**

stand alone:

```
DoMethod(txobj,MUIM_SetAsString,MUIA_Text_Contents,
        "My name is %s and I am %ld years old.",name,age);
```

within a notification statement:

```
DoMethod(propobj,MUIM_Notify,MUIA_Prop_First,MUIV_EveryTime,
        txobj,4,MUIM_SetAsString,MUIA_Text_Contents,
        "prop gadget shows %ld.",MUIV_TriggerValue);
```

**SEE ALSO**

MUIM\_Set, MUIM\_Notify

**22.10 MUIM\_SetUData (V8)****SYNOPSIS**

```
DoMethod(obj,MUIM_SetUData,ULONG udata, ULONG attr, ULONG val);
```



**FUNCTION**

This method tests if the MUIA\_UserData of the object contains the given <udata> and sets <attr> to <val> for itself in this case.

Although this is not very useful for single objects, performing this method on objects that handle children can become very handy. In this case, all the children (any maybe their children) are tested against <udata> and all matching objects will get the attribute set.

If you e.g. want to clear several string gadgets in your application at once, you simply give them the same MUIA\_UserData and use

```
DoMethod(app,MUIM_SetUData,MyUDATA,MUIA_String_Contents,NULL);
```

**INPUTS**

**udata** - userdata to look for.

**attr** - attribute to set.

**val** - value to set attribute to.

**NOTE**

If you have many objects in your application, MUIM\_SetUData may take quite long. You can limit the amount of time by performing the method not on the application but on the window or even on the group your gadgets are place in.

**SEE ALSO**

MUIM\_GetUData, MUIM\_FindUData

**22.11 MUIM\_WriteLong (V6)****SYNOPSIS**

```
DoMethod(obj,MUIM_WriteLong,ULONG val, ULONG *memory);
```

**FUNCTION**

This method simply writes a longword somewhere to memory. Although this seems quite useless, it might become handy if used within a notify statement. For instance, you could easily connect the current level of a slider with some member of your programs data structures.

**INPUTS**

**val** - value to write

**memory** - location to write the value to

**EXAMPLE**

```
/* Let the slider automagically write its level to a variable */

static LONG level;

DoMethod(slider,MUIM_Notify,MUIA_Slider_Level,MUIV_EveryTime,
    slider,3,MUIM_WriteLong,MUIV_TriggerValue,&level);
```

**SEE ALSO**

MUIM\_WriteString, MUIM\_Notify

**22.12 MUIM\_WriteString (V6)****SYNOPSIS**

```
DoMethod(obj,MUIM_WriteString,char *str, char *memory);
```

**FUNCTION**

This method simply copies a string somewhere to memory. Although this seems quite useless, it might become handy if used within a notify statement. For instance, you could easily connect the current contents of a string gadget with some member of your programs data structures.

**NOTE**

The string is copied with strcpy(), you must assure that the destination points to enough memory.

**INPUTS**

**str** - string to copy

**memory** - location to write the value to

**EXAMPLE**

```
static char buffer[256];

DoMethod(string,MUIM_Notify,MUIA_String_Contents,MUIV_EveryTime,
    string,3,MUIM_WriteString,MUIV_TriggerValue,buffer);
```

**SEE ALSO**

MUIM\_WriteLong, MUIM\_Notify

**22.13 MUIA\_AppMessage – (V5) [..G], struct AppMessage \*****FUNCTION**

When your window is an AppWindow, i.e. you have set the MUIA\_Window\_AppWindow attribute to TRUE, you will be able to get AppMessages by listening to MUIA\_AppMessage. Whenever an AppMessage arrives, this attribute will be set to a pointer to that message.

MUIA\_AppMessage is object specific. You can e.g. set up different notifications for different objects in your window, they will only get executed when icons are dropped over the specific object.

If you wait on MUIA\_AppMessage with a window object, your notify will always get executed when icons are dropped on the window.

**NOTE**

- You should use the MUIM\_CallHook method to call a hook function when an AppMessage arrives (see below). The pointer to the AppMessage is valid only as long as the notification method is executed.
- AppWindows are only possible on the workench screen.

**EXAMPLE**

```
/* Call the AppMsgHook when an icon is dropped on a listview */
DoMethod(lvobj,MUIM_Notify,MUIA_AppMessage,MUIV_EveryTime,
         lvobj,3,MUIM_CallHook,&AppMsgHook,MUIV_TriggerValue);

/* Call the AppMsgHook when an icon is dropped on the window */
DoMethod(winobj,MUIM_Notify,MUIA_AppMessage,MUIV_EveryTime,
         winobj,3,MUIM_CallHook,&AppMsgHook,MUIV_TriggerValue);
```

**SEE ALSO**

MUIA\_Window\_AppWindow, MUIA\_Application\_DropObject, MUIM\_CallHook

## 22.14 MUIA\_HelpFile – (V4) [ISG], STRPTR (OBSOLETE) FUNCTION

Since muimaster.library V8, this attribute is obsolete and replaced by MUIA\_Application\_HelpFile.

**SEE ALSO**

MUIA\_Application\_HelpFile, MUIA\_HelpNode, MUIA\_HelpLine

## 22.15 MUIA\_HelpLine – (V4) [ISG], LONG FUNCTION

Define a line in a help file specified with MUIA\_Application\_HelpFile.

**SEE ALSO**

MUIA\_Application\_HelpFile, MUIA\_HelpNode

## 22.16 MUIA\_HelpNode – (V4) [ISG], STRPTR FUNCTION

Define a node in a help file specified with MUIA\_Application\_HelpFile.

**SEE ALSO**

MUIA\_Application\_HelpFile, MUIA\_HelpLine

## 22.17 MUIA\_NoNotify – (V7) [.S.], BOOL

### FUNCTION

If you set up a notify on an attribute to react on user input, you will also recognize events when you change this attribute under program control with `SetAttrs()`. Setting `MUIA_NoNotify` together with your attribute will prevent this notification from being triggered.

### NOTE

`MUIA_NoNotify` is a "one time" attribute. Its only valid during the current `SetAttrs()` call!

### EXAMPLE

```
SetAttrs(slider,MUIA_NoNotify,TRUE,MUIA_Slider_Level,26,TAG_DONE);
```

## 22.18 MUIA\_Revision – (V4) [..G], LONG

### FUNCTION

Get the revision number of an objects class. Although `MUIA_Revision` is documented at notify class, you will of course receive the revision number of the objects true class.

### EXAMPLE

```
strobj = MUI_NewObject(MUIC_String,...,TAG_DONE);
...
get(strobj,MUIA_Version,&v);
get(strobj,MUIA_Revision,&r);
printf("String class version %ld.%ld\n",v,r);
```

### SEE ALSO

`MUIA_Version`

## 22.19 MUIA\_UserData – (V4) [ISG], ULONG

### FUNCTION

A general purpose value to fill in any kind of information.

## 22.20 MUIA\_Version – (V4) [..G], LONG

### FUNCTION

Get the version number of an objects class. Although `MUIA_Version` is documented at notify class, you will of course receive the version number of the objects true class.

**EXAMPLE**

```
strobj = MUI_NewObject(MUIC_String,...,TAG_DONE);
...
get(strobj,MUIA_Version ,&v);
get(strobj,MUIA_Revision,&r);
printf("String class version %ld.%ld\n",v,r);
```

**SEE ALSO**

MUIA\_Revision

**23 Palette.mui**

Palette class generates a (big) group of objects, altogether making up a powerful palette requester. Due to the new color selection schemes of Kickstart 3.x, you won't get a "traditional" palette requester with 256 fields to fill in. These things really stop making sense on nice 256 or true color screens.

Instead, MUI's palette class allows defining a list of colors that the user should be able to adjust. Within a public screen manager, this would e.g. be the DrawInfo pens for a specific screen, within a terminal program maybe the eight ANSI colors.

Palette class uses a listview to let the user choose the desired color, a coloradjust object to adjust this color and a colorfield object that always shows the current color.

The user will also be able to concatenate several colors in the list, defining a single color for several entries.

### **23.1 MUIA\_Palette\_Entries – (V6) [I.G], struct MUI\_Palette\_Entry \***

**FUNCTION**

Specify the colors that the user should be able to adjust with this palette object.

You supply an array of MUI\_Palette\_Structures here, each entry defining one color:

```
struct MUI_Palette_Entry
{
    LONG   mpe_ID;
    ULONG  mpe_Red;
    ULONG  mpe_Green;
    ULONG  mpe_Blue;
    LONG   mpe_Group;
};
```

**mpe\_ID** This entry is not used by palette class, you can put in whatever you want, except the value MUIV\_Palette\_Entry\_End (==1), which terminates the array.

**mpe\_Red** 32-bit red component of the current color. This field will be changed by palette class whenever the user edits the color.

**mpe\_Green** 32-bit green component of the current color. This field will be changed by palette class whenever the user edits the color.

**mpe\_Blue** 32-bit blue component of the current color. This field will be changed by palette class whenever the user edits the color.

**mpe\_Group** Entries with the same **mpe\_Group** value are concatenated. Whenever a new color in the listview is selected, all other colors with the same **mpe\_Group** get selected as well and get adjusted all at once. Entry concatenation can be changed by the user, as long as you don't disable this feature with the **MUIA\_Palette\_Groupable** attribute.

### EXAMPLE

```
static struct MUI_Palette_Entry SystemDefaultPalette[] =
{
    { TEXTPEN           ,0x00000000,0x00000000,0x00000000,0 },
    { SHINEPEN          ,0xffffffff,0xffffffff,0xffffffff,1 },
    { SHADOWPEN         ,0x00000000,0x00000000,0x00000000,0 },
    { FILLPEN           ,0x66666666,0x88888888,0xbbbbbbbb,2 },
    { FILLTEXTPEN       ,0xffffffff,0xffffffff,0xffffffff,1 },
    { BACKGROUNDPEN    ,0xaaaaaaaa,0xaaaaaaaa,0xaaaaaaaa,3 },
    { HIGHLIGHTTEXTPEN ,0xffffffff,0xffffffff,0xffffffff,1 },
    { BARDETAILPEN      ,0x00000000,0x00000000,0x00000000,0 },
    { BARBLOCKPEN       ,0xffffffff,0xffffffff,0xffffffff,1 },
    { BARTRIMPEN        ,0x00000000,0x00000000,0x00000000,0 },
    { MUIV_Palette_Entry_End,0,0,0,0 },
};
```

### SEE ALSO

**MUIA\_Palette\_Names**

## 23.2 MUIA\_Palette\_Groupable – (V6) [ISG], BOOL

### FUNCTION

Enables/disables palette color grouping. Defaults to TRUE.

### SEE ALSO

**MUIA\_Palette\_Entries**

## 23.3 MUIA\_Palette\_Names – (V6) [ISG], char \*\*

### FUNCTION

Specify the names of a palette objects color entries. Without names, the color listview just displays "Color <n>" for each entry. If you supply an array of names here, they are displayed instead. The names array must have as many entries as the array of **MUIA\_Palette\_Entry** structures (without its terminator).

### EXAMPLE

```
static struct MUI_Palette_Entry ColorEntries[] =
{
    { TEXTPEN           ,0x00000000,0x00000000,0x00000000,2 },
    { SHINEPEN          ,0xffffffff,0xffffffff,0xffffffff,4 },
    { SHADOWPEN         ,0x00000000,0x00000000,0x00000000,5 },
    { FILLPEN           ,0x66666666,0x88888888,0xbbbbbbbb,3 },
    { FILLTEXTPEN       ,0xffffffff,0xffffffff,0xffffffff,6 },
};
```

```

    { BACKGROUNDPEN      ,0x00000000,0x00000000,0x00000000,7 },
    { HIGHLIGHTTEXTPEN,0xffffffff,0xffffffff,0xffffffff,8 },
    { BARDETAILPEN       ,0x00000000,0x00000000,0x00000000,9 },
    { BARBLOCKPEN        ,0xffffffff,0xffffffff,0xffffffff,1 },
    { BARTRIMPEN         ,0x00000000,0x00000000,0x00000000,0 },
    { MUIV_Palette_Entry_End,0,0,0,0 },
};

static const char *ColorNames[] =
{
    "Text"                ,
    "Bright Edges"        ,
    "Dark Edges"          ,
    "Active Window Bars" ,
    "Active Window Titles",
    "Background"          ,
    "Important Text"      ,
    "Menu Text"           ,
    "Menu Background"     ,
    "Menu Line"
};

po = PaletteObject,
    MUIA_Palette_Entries, ColorEntries,
    MUIA_Palette_Names   , ColorNames,
    End;

```

## SEE ALSO

MUIA\_Palette\_Entries

## 24 Popasl.mui

As a subclass of popstring class, popasl can be used to pop up any kinds of standard system asl requesters. A separate task is spawned to handle these requesters, the application continues to run.

Using an asl popup class, you don't need to worry about handling asl requesters. MUI will automatically open one when the popup button is pressed and update the corresponding string gadget when the user terminates the requester. From the programmers point of view, all you have to do is to handle the string gadgets contents.

IMPORTANT: At object creation time, you can use all ASL library tags as well. They will be passed to the AllocAslRequest() call without further interpretation.

### 24.1 MUIA\_Popasl\_Active – (V7) [..G], BOOL

#### FUNCTION

Popasl creates asynchronous popups. Requesters are opened in a separately spawned task and don't disturb the rest of the application. You can ask for the state of a requester by querying the MUIA\_Popasl\_Active attribute. It will return TRUE when the requester is currently open, FALSE otherwise.

Common use for this attribute is to prevent an application from being terminated while a requester is open. If you try to dispose the popasl object with a currently open requester, MUI will freeze your task as long as the requester stays there.

**EXAMPLE**

```

case MUIV_Application_ReturnID_Quit:
{
    LONG active;

    get(pop1,MUIA_Popasl_Active,&active);
    if (!active) get(pop2,MUIA_Popasl_Active,&active);
    if (!active) get(pop3,MUIA_Popasl_Active,&active);
    if (!active) get(pop4,MUIA_Popasl_Active,&active);

    if (active)
        MUI_Request(app>window,0,NULL,"OK",
                    "Cannot quit now, still some asl popups opened.");
    else
        running = FALSE;
}
break;

```

**SEE ALSO**

MUIA\_Popasl\_StartHook, MUIA\_Popasl\_StopHook, MUIA\_Popasl\_Type

## 24.2 MUIA\_Popasl\_StartHook – (V7) [ISG], struct Hook \*

### FUNCTION

Before popasl class opens the asl requester, it has to get some kind of parameters describing its initial contents. A file popup would e.g. need to split the string gadgets contents into path and file name part and pass these as ASLFR\_InititalFile and ASLFR\_InitialDrawer to the requester.

The MUIA\_Popasl\_StartHook tag describes a hook function that will be called immediately before the requester is opened. It will receive a pointer to itself in A0, a pointer to the popasl object in A2 and a pointer to a taglist in A1. This taglist already contains some tags:

ASLFR/FO/...Screen	:	parent screen
ASLFR/FO/..._PrivateIDCMP	:	TRUE
ASLFR/FO/..._InititalLeftEdge	:	left edge of popasl object
ASLFR/FO/..._InititalTopEdge	:	bottom edge of popasl object
ASLFR/FO/..._InititalWidth	:	width of popasl object, only present when the popup is called for the first time.

You may add other tags to the list, but beware that the maximum allowed number of tags is 15. If you need more, use the TAG\_MORE tag.

Since the asl requester will run in a separate task, you should not change the state of the ASLFR\_PrivateIDCMP tag!

If your hook returns TRUE, popasl class opens the requester with the given taglist. A return value of FALSE should be used when something went wrong, no requester will be opened in this case.

For file and font requester, popasl class will fall back to a default tag handling when no start hook is specified. A file name is automatically split into path and file part and passed to the requester a ASLFR\_InitialFile and ASLFR\_InitialDrawer. A font requester splits a string like "topaz/8" into font name and size for ASLFO\_InitialName and ASLFO\_InitialSize.



**SEE ALSO**

MUIA\_Popasl\_StopHook, MUIA\_Popasl\_Type

### 24.3 MUIA\_Popasl\_StopHook – (V7) [ISG], struct Hook \*

#### FUNCTION

When the requester terminates, MUIA\_Popasl\_StopHook will be called with a pointer to itself in A0, a pointer to the popasl object in A2 and a pointer to the asl requester structure in A1. The hook can then parse the requester structure and set the string gadgets contents respectively.

For file and font requesters, a default handling is provided.

**SEE ALSO**

MUIA\_Popasl\_StartHook, MUIA\_Popasl\_Type

### 24.4 MUIA\_Popasl\_Type – (V7) [I.G], ULONG

#### FUNCTION

This tag allows to set the type of asl requester. Pass the same value you would use for AllocAslRequest(), e.g. ASL\_FileRequest, ASL\_FontRequest or ASL\_ScreenModeRequest.

For ASL\_FileRequest and ASL\_FontRequest, popasl class offers a a standard start/stop handling. When a file requester is opened, MUI splits the string gadgets contents into a path and a file name and uses these as initial paremeters for the requester. Font popups translate a font into a name/size pair, e.g. "topaz/8". You can override these translations by specifying a MUIA\_Popasl\_StartHook and a MUIA\_Popasl\_StopHook.

For ASL\_ScreenModeRequest, no standard handling is available. Using such a popup without Start and Stop hooks won't make much sense.

**SEE ALSO**

MUIA\_Popasl\_StartHook, MUIA\_Popasl\_StopHook

## 25 Poplist.mui

Poplist class simplifies creation of popups that contain just a simple list of predefined gadget contents.

### 25.1 MUIA\_Poplist\_Array – (V8) [I.], char \*\*

#### FUNCTION

A NULL terminated list of strings defining the contents of the poplist object.

## 26 Popobject.mui

Popobject class takes a MUI object as parameter uses this one as popup. You can e.g. simply create a listview object with some entries and the popobject class will create a window around it and display it when the user hits the popup button.

Using this class instead of creating the popup windows yourself prevents you from having lots of problems. Think twice before deciding to make you own popups!

### 26.1 MUIA\_Popobject\_Follow – (V7) [ISG], BOOL

#### FUNCTION

Setting this attribute causes the popup window to follow its parent window when its moved. Defaults to TRUE.

#### SEE ALSO

MUIA\_Popobject\_Light, MUIA\_Popobject\_Volatile.

### 26.2 MUIA\_Popobject\_Light – (V7) [ISG], BOOL

#### FUNCTION

This attribute causes the popup window to be border and titleless. Defaults to TRUE

#### SEE ALSO

MUIA\_Popobject\_Follow, MUIA\_Popobject\_Volatile

### 26.3 MUIA\_Popobject\_Object – (V7) [I.G], Object \*

#### FUNCTION

Specify the object to pop up. Usually this is a relatively simple thing like a single listview, but you can of course use group class here and make rather complex popups. As with all other MUI classes, the object here gets disposed when the popobject is disposed.

#### EXAMPLE

```
pop = PopobjectObject,
    MUIA_Popstring_String, KeyString(0,60,'n'),
    MUIA_Popstring_Button, PopButton(MUII_PopUp),
    MUIA_Popobject_StrObjHook, &StrObjHook,
    MUIA_Popobject_ObjStrHook, &ObjStrHook,
    MUIA_Popobject_Object, ListviewObject,
        MUIA_Listview_List, ListObject,
            InputListFrame,
            MUIA_List_SourceArray, PopNames,
            End,
        End,
    End;
```

#### SEE ALSO

MUIA\_Popobject\_StrObjHook, MUIA\_Popobject\_ObjStrHook,  
MUIA\_Popobject\_Light

### 26.4 MUIA\_Popobject\_ObjStrHook – (V7) [ISG], struct Hook \*

#### FUNCTION

When a popup is closed, this hook is called. You can examine the state of your MUIA\_Popobject\_Object and set the contents of the string gadget respectively. The

hook receives a pointer to itself in A0, a pointer to your MUIA\_Popobject\_Object in A2 and a pointer to the embedded string object in A1.

The hook will only be called when your popup is closed with a success value of TRUE. Otherwise, MUI closes the popup without taking further actions, just as if had never opened.

Since MUI doesn't know anything about your MUIA\_Popobject\_Object, it's your task to tell when your popup is finished. You can terminate popups at anytime by sending a MUIM\_Popstring\_Close method:

**A double click terminates the popping list with a successful return value.**

```
DoMethod(plist,MUIM_Notify,MUIA_Listview_DoubleClick,TRUE,
pop,2,MUIM_Popstring_Close,TRUE);
```

#### EXAMPLE

```
SAVEDS ASM VOID ObjStrFunc(REG(a2) Object *list,REG(a1) Object *str)
{
    char *x;
    DoMethod(list,MUIM_List_GetEntry,MUIV_List_GetEntry_Active,&x);
    set(str,MUIA_String_Contents,x);
}
```

## 26.5 MUIA\_Popobject\_StrObjHook – (V7) [ISG], struct Hook \*

#### FUNCTION

Before the popup opens, this hook is called. You can use it to prepare your MUIA\_Popobject\_Object according to the contents of the string gadget. The hook receives a pointer to itself in A0, a pointer to your MUIA\_Popobject\_Object in A2 and a pointer to the embedded string object in A1.

Return TRUE if you want the popup to appear, FALSE otherwise.

#### EXAMPLE

```
SAVEDS ASM LONG StrObjFunc(REG(a2) Object *list,REG(a1) Object *str)
{
    char *x,*s;
    int i;

    get(str,MUIA_String_Contents,&s);

    for (i=0;;i++)
    {
        DoMethod(list,MUIM_List_GetEntry,i,&x);
        if (!x)
        {
            set(list,MUIA_List_Active,MUIV_List_Active_Off);
            break;
        }
        else if (!strcmp(x,s))
        {
            set(list,MUIA_List_Active,i);
            break;
        }
    }
}
```

```
    return(TRUE);
}
```

### SEE ALSO

MUIA\_Popobject\_ObjStrHook, MUIA\_Popobject\_Object,  
MUIA\_Popobject\_WindowHook

## 26.6 MUIA\_Popobject\_Volatile – (V7) [ISG], BOOL

### FUNCTION

Setting this attribute causes the popup window to disappear when the corresponding popobject disappears, e.g. because its in a page group and the user toggled the page. When the popobject appears again, the popup window appears also. Defaults to TRUE.

### SEE ALSO

MUIA\_Popobject\_Light, MUIA\_Popobject\_Follow

## 26.7 MUIA\_Popobject\_WindowHook – (V9) [ISG], struct Hook \*

### FUNCTION

If specified, this hook is called immediately after the popups window objects has been created but before this window is opened. You might e.g. want to add a cycle chain for the popup window here.

The hook is called with a pointer to the pop object (MUIA\_Popobject\_Object) in A2 and with a pointer to the window object that MUI generated to handle the popup in A1.

### EXAMPLE

```
/* pop is a simple listview, just set the windows
** default object to this to enable keyboard control */

SAVEDS ASM VOID WindowFunc(REG(a2) Object *pop,REG(a1) Object *win)
{
    set(win,MUIA_Window_DefaultObject,pop);
}
```

### SEE ALSO

MUIA\_Popobject\_ObjStrHook, MUIA\_Popobject\_Object

## 27 Popstring.mui

Popstring class is the base class for creating so called popup objects. Usually, a popup consists of a string or text gadget, followed by a little button. Pressing this button brings up a little window with a listview and lets the user choose an entry with the mouse.

Popstring class features the basic functions for creating such objects. Given a string object and a button object, it places them horizontally and sets up some notification. Whenever the popup button is pressed, a hook will be called which itself should open and prepare the popup window.

The string and the button object are not created by popstring class, they have to be supplied as attributes during object creation time. This makes popstring class very flexible, one could e.g. use a text object instead of a string or a popup button with some text in it.

However, creating simple popups with popstring class would be too much overhead. Instead of using it directly, you should have a look at one of its subclasses. They offer a more specialized set of popups and are a lot easier to use.

## 27.1 MUIM\_Popstring\_Close (V7)

### SYNOPSIS

```
DoMethod(obj,MUIM_Popstring_Close,LONG result);
```

### FUNCTION

This method closes the popup. In fact, it only calls the predefined MUIA\_Popstring\_CloseHook with the supplied success parameter.

### EXAMPLE

```
DoMethod(poplist,MUIM_Notify,MUIA_Listview_DoubleClick,TRUE,
    popobj,2,MUIM_Popstring_Close,TRUE);
```

## 27.2 MUIM\_Popstring\_Open (V7)

### SYNOPSIS

```
DoMethod(obj,MUIM_Popstring_Open,);
```

### FUNCTION

This method opens the popup. In fact, it only calls the predefined MUIA\_Popstring\_OpenHook and checks its return value. In case of TRUE, the popup button object is disabled as long as MUIA\_Popstring\_Toggle is unset.

If the toggle mode is enabled, using MUIA\_Popstring\_Open on a currently opened popup will result in closing this popup (i.e. calling the close hook) with a success value of FALSE.

### EXAMPLE

```
DoMethod(popbutton,MUIM_Notify,MUIA_Pressed,FALSE,
    popobj,1,MUIM_Popstring_Open);
```

## 27.3 MUIA\_Popstring\_Button – (V7) [I.G], Object \*

### FUNCTION

Specify the button object to be used in the popup. Depending on the type of your popup, you should use an image button with MUII\_PopUp, MUII\_PopFile

or MUII\_PopDrawer here. However, its also possible to have a button with some text or other things in it.

When the popstring object is disposed, the string and the button objects are disposed as well.

### EXAMPLE

```
pop = PopstringObject,
    MUIA_Popstring_String, KeyString(0,60,'n'),
    MUIA_Popstring_Button, PopButton(MUII_PopUp),
    MUIA_Popstring_OpenHook, &OpenHook,
    MUIA_Popstring_CloseHook, &CloseHook,
    End;
```

### SEE ALSO

MUIA\_Popstring\_String, MUIA\_Popstring\_OpenHook, MUIA\_Popstring\_CloseHook

## 27.4 MUIA\_Popstring\_CloseHook – (V7) [ISG], struct Hook

\*

### FUNCTION

Whenever the popup receives a MUIM\_Popstring\_Close method and the popup is currently opened, this hook will be called. It will receive a pointer to itself in register A0, a pointer to the complete popup object in A2 and a pointer to a

```
struct
{
    Object *stringobject;
    LONG success;
}
```

in A1. The success parameter is a copy of the methods success parameter and indicates whether the popup was closed successfully (e.g. with a double click in a listview) or was just cancelled (e.g. by pressing the popup button again for toggle popups).

Due to internal message handling issues, calling the close hook is delayed until the next MUIM\_HandleInput method is called. This allows you to remove and dispose windows without danger.

### SEE ALSO

MUIA\_Popstring\_OpenHook, MUIM\_Popstring\_Open, MUIM\_Popstring\_Close

## 27.5 MUIA\_Popstring\_OpenHook – (V7) [ISG], struct Hook

\*

### FUNCTION

Whenever the popup receives a MUIM\_Popstring\_Open method, this hook will be called. It will receive a pointer to itself in register A0, a pointer to the complete popup object in A2 and a pointer to a pointer (!) to the string object contained in the popup object in A1.

When this hook returns TRUE, MUI assumes the popup was opened successfully and will disabled the popup button (as long as MUIA\_Popstring\_Toggle is not set).

Return FALSE to indicate that something went wrong and the popup could not be opened.

#### SEE ALSO

MUIA\_Popstring\_CloseHook, MUIM\_Popstring\_Open, MUIM\_Popstring\_Close

### 27.6 MUIA\_Popstring\_String – (V7) [I.G], Object \*

#### FUNCTION

Specify the string object to be used in the popup. This does not necessarily need to be a real string object, using text objects or even complete groups of other objects is perfectly ok.

When the popstring object is disposed, the string and the button objects are disposed as well.

#### EXAMPLE

```
pop = PopstringObject,
    MUIA_Popstring_String, KeyString(0,60,'n'),
    MUIA_Popstring_Button, PopButton(MUII_PopUp),
    MUIA_Popstring_OpenHook, &OpenHook,
    MUIA_Popstring_CloseHook, &CloseHook,
    End;
```

#### SEE ALSO

MUIA\_Popstring\_Button, MUIA\_Popstring\_OpenHook,  
MUIA\_Popstring\_CloseHook

### 27.7 MUIA\_Popstring\_Toggle – (V7) [ISG], BOOL

#### FUNCTION

Set/Clear the toggle mode for a popstring object. With toggling disabled, the popup button will get disabled whenever the user hits it and the popup opens. With toggling enabled, the popup button always stays enabled and can be used to cancel (== close with a FALSE return value) the popup.

#### SEE ALSO

MUIA\_Popstring\_OpenHook

## 28 Prop.mui

Prop class generates the well known proportional gadgets. It offers the same attributes as a usual boopsi gadget of progclass. However, MUI's prop gadgets allow using any imagery for the knob and for the background.

### 28.1 MUIA\_Prop\_Entries – (V4) [ISG], LONG

#### FUNCTION

Set or get the total number of entries.

**SEE ALSO**

MUIA\_Prop\_Horiz, MUIA\_Prop\_Visible, MUIA\_Prop\_First

### **28.2 MUIA\_Prop\_First – (V4) [ISG], LONG FUNCTION**

Set or get the number of the first entry.

**SEE ALSO**

MUIA\_Prop\_Horiz, MUIA\_Prop\_Visible, MUIA\_Prop\_Entries

### **28.3 MUIA\_Prop\_Horiz – (V4) [I.G], BOOL FUNCTION**

Determine if you want a horizontal or a vertical prop gadget.  
Defaults to FALSE, i.e. vertical.

**SEE ALSO**

MUIA\_Prop\_Entries, MUIA\_Prop\_Visible, MUIA\_Prop\_First

### **28.4 MUIA\_Prop\_Slider – (V4) [ISG], BOOL FUNCTION**

Indicate that this prop gadget is used in a slider. MUI might then use different imagery. Since you really should use the slider class when creating sliders, you normally don't need to care about this attribute.

### **28.5 MUIA\_Prop\_Visible – (V4) [ISG], LONG FUNCTION**

Set or get the number of visible entries.

**SEE ALSO**

MUIA\_Prop\_Horiz, MUIA\_Prop\_Entries, MUIA\_Prop\_First

## **29 Radio.mui**

Radio class generates radio button gadgets. They do the same job as cycle gadgets and eat up more window space, maybe that's the reason why so few of them can be found in existing applications.

### **29.1 MUIA\_Radio\_Active – (V4) [ISG], LONG FUNCTION**

This attributes defines the number of the active entry in the radio gadgets. Valid range is from 0 for the first entry to NumEntries-1 for the last.

Setting MUIA\_Radio\_Active causes the gadget to be updated. On the other hand, when the user plays around with the gadget, MUIA\_Radio\_Active will always reflects the current state.



**EXAMPLE**

```
set(radioobj,MUIA_Radio_Active,3);
```

**SEE ALSO**

MUIA\_Radio\_Entries

## 29.2 MUIA\_Radio\_Entries – (V4) [I.], STRPTR \*

### FUNCTION

Here you can define what entries shall be displayed in your radio gadget. You must supply a pointer to a string array, containing one entry for each item and terminated with a NULL.

Remember that radio gadget entries may contain any text formatting code such as bold, italic or underlined characters.

**EXAMPLE**

```
static const char *RA_GroupTitleColor[] =
{
    "normal",
    "highlight",
    "3-dimensional",
    NULL
};

CY_Title = RadioObject,
MUIA_Radio_Entries, RA_GroupTitleColor,
End;
```

**SEE ALSO**

MUIA\_Radio\_Active, MUIA\_Text\_Contents

## 30 Rectangle.mui

Rectangle class seems kind of useless since it doesn't define any attributes or methods itself. However, objects of this type are frequently used in every application. They allow insertion of space to control MUI's layout process.

### 30.1 MUIA\_Rectangle\_HBar – (V7) [I.G], BOOL

**FUNCTION**

When set to TRUE, MUI draws a horizontal bar in the middle of the rectangle. Such bars can be used instead of group frames to separate objects in a window.

**EXAMPLE**

```
/* draw a two pixel high bar in the middle
   of an 8 pixel high rectangle */

RectangleObject, MUIA_Rectangle_HBar, TRUE, MUIA_FixHeight, 8, End;
```

**SEE ALSO**

MUIA\_Rectangle\_VBar

**30.2 MUIA\_Rectangle\_VBar – (V7) [I.G], BOOL****FUNCTION**

When set to TRUE, MUI draws a vertical bar in the middle of the rectangle. Such bars can be used instead of group frames to separate objects in a window.

**EXAMPLE**

```
/* draw a two pixel wide bar in the middle
   of an 8 pixel wide rectangle */
```

```
RectangleObject, MUIA_Rectangle_HBar, TRUE, MUIA_FixWidth, 8, End;
```

**SEE ALSO**

MUIA\_Rectangle\_HBar

**31 Register.mui**

Register class is a special class for handling multi page groups. Using this class, you only have to supply an array of strings, describing the children's titles. How these titles are visualized, either with a cycle gadget or with a register-like group, is the choice of the user.

**31.1 MUIA\_Register\_Frame – (V7) [I.G], BOOL****FUNCTION**

Specify TRUE if you want your group to be framed. If the user specified cycle gadget looking, you will get a group frame, otherwise you won't get any frame at all since register groups are framed anyway.

**SEE ALSO**

MUIA\_Register\_Titles

**31.2 MUIA\_Register\_Titles – (V7) [I.G], STRPTR \*****FUNCTION**

NULL terminated array of strings describing the titles of your groups children. This array must contain exactly as many as entries as your group has children.

**EXAMPLE**

```
static const char *titles[] = { "Eyes", "Ears", "Noses", "Feet", NULL };

obj = RegisterGroup(title),
    Child, ...,
    Child, ...,
    Child, ...;
```

```
Child, ...,
End;
```

## 32 Scale.mui

A Scale object generates a percentage scale running from 0% to 100%. A good place for such an object is e.g. below a fuel gauge.

Depending on how much space is available, the scale will be more or less detailed.

Due to MUI's automatic layout system, you don't need to worry about its size. When placed in a vertical group just below the object you want to scale, everything is fine.

### 32.1 MUIA\_Scale\_Horiz – (V4) [ISG], BOOL

#### FUNCTION

Indicate whether you want a horizontal or a vertical scale.

Defaults to horizontal.

BUGS Currently, only the horizontal scale is implemented.

#### EXAMPLE

```
...
VGroup,
    Child, GaugeObject, End,
    Child, Scaleobject, End,
    End,
...
/* and everythings is fine... */
```

## 33 Scrmodelist.mui

This is a private class and only used by the MUI preferences program. Maybe it will get public in a future release.

## 34 Scrollbar.mui

The Scrollbar class has no objects and attributes itself. It just connects a proportional gadget and two button gadgets with appropriate imagery to make up a scrollbar.

Since Scrollbar class is a subclass of Group class, every attribute and method is passed through to all of its children. Thus, you can talk and listen to a scrollbar as if it was just a single prop gadget.

You can use the attribute MUIA\_Group\_Horiz as with any other group to determine if the scrollbar should be horizontal or vertical. By default, a vertical scrollbar is generated.

## 35 Scrollgroup.mui

Scrollgroup objects can be used to supply virtual groups with scrollbars. These scrollbars automatically adjust according to the virtual and display sizes of the

underlying virtual group. When scrolling is unnecessary (i.e. the virtual group is completely visible), the scrollers get disabled.

### 35.1 MUIA\_Scrollgroup\_Contents – (V4) [I.], Object \*

#### FUNCTION

You have to specify an object of Virtgroup class here.

### 35.2 MUIA\_Scrollgroup\_FreeHoriz – (V9) [I.], BOOL

#### FUNCTION

Specify if a scroll group should be horizontally moveable. Defaults to FALSE.

### 35.3 MUIA\_Scrollgroup\_FreeVert – (V9) [I.], BOOL

#### FUNCTION

Specify if a scroll group should be vertically moveable. Defaults to FALSE.

## 36 Slider.mui

The slider class generates a gui element that allows a user to adjust a numeric value. The programmer has not very much influence on the slider's outfit, there are only very few tags available. Future versions of MUI will probably include some preferences options to allow the user (**not** the programmer) to configure this outfit.

Note that since slider is a subclass of group class, you can get horizontal or vertical sliders by simply using the MUIA\_Group\_Horiz attribute. Default is a horizontal slider.

### 36.1 MUIA\_Slider\_Level – (V4) [ISG], LONG

#### FUNCTION

The current position of the slider knob. This value is guaranteed to be between MUIA\_Slider\_Min and MUIA\_Slider\_Max.

#### EXAMPLE

```
/* vertical task priority slider */
SliderObject,
    MUIA_Group_Horiz , FALSE,
    MUIA_Slider_Min   , -20,
    MUIA_Slider_Max   ,  20,
    MUIA_Slider_Level,   0,
End;
```

#### SEE ALSO

MUIA\_Slider\_Min, MUIA\_Slider\_Max

### 36.2 MUIA\_Slider\_Max – (V4) [ISG], LONG

#### FUNCTION

Adjust the maximum value for a slider object.

**SEE ALSO**

MUIA\_Slider\_Min, MUIA\_Slider\_Level

**36.3 MUIA\_Slider\_Min – (V4) [ISG], LONG****FUNCTION**

Adjust the minimum value for a slider object. Of course you can use negative number, e.g. for a slider to adjust task priority.

**SEE ALSO**

MUIA\_Slider\_Max, MUIA\_Slider\_Level

**36.4 MUIA\_Slider\_Quiet – (V6) [I.], BOOL****FUNCTION**

When set to TRUE, the slider doesn't display it's current level in a text object.

**SEE ALSO**

MUIA\_Slider\_Level

**36.5 MUIA\_Slider\_Reverse – (V4) [ISG], BOOL****FUNCTION**

Setting this attribute to TRUE will reverse the direction of the slider.

**SEE ALSO**

MUIA\_Slider\_Min, MUIA\_Slider\_Max, MUIA\_Slider\_Level

**37 String.mui**

String class generates standard string gadgets with all editing facilities (clear, undo, etc.) enabled.

**37.1 MUIA\_String\_Accept – (V4) [ISG], STRPTR****FUNCTION**

A string containing characters allowed as input for the string gadget. Whenever the user hits a character not found in MUIA\_String\_Accept, he will hear a beep and gadgets contents won't have changed.

**EXAMPLE**

```
StringObject,
    MUIA_String_Accept, "0123456789-",
    End,
```

**SEE ALSO**

MUIA\_String\_Reject

### 37.2 MUIA\_String\_Acknowledge – (V4) [..G], STRPTR

#### FUNCTION

This attribute will be set to the contents of the string whenever the user hits return in the gadget. An application can listen with notification and take the appropriate action.

Using the TAB key or a mouse click to deactivate the gadget will not trigger MUIA\_String\_Acknowledge.

#### EXAMPLE

```
/* two string gadgets str1 and str2, the second should
/* become active after a return in the first: */

DoMethod(str1,MUIM_Notify,
    MUIA_String_Acknowledge, MUIV_EveryTime,
    windowobj, 3, MUIM_Set, MUIA_Window_ActiveObject, str2);
```

#### SEE ALSO

MUIA\_String\_Contents

### 37.3 MUIA\_String\_AttachedList – (V4) [I..], Object \*

#### FUNCTION

This special attribute can be set to point to a valid MUI object of List or Listview class. This enables controlling the lists cursor from within the string gadget, all cursor key events will be forwarded.

#### SEE ALSO

MUIA\_String\_Contents, MUIA\_List\_Active

### 37.4 MUIA\_String\_BufferPos – (V4) [..SG], LONG

#### FUNCTION

MUIA\_String\_BufferPos can be used to get and set the position of the cursor in the string gadget. This attribute is probably not very interesting.

#### SEE ALSO

MUIA\_String\_Contents, MUIA\_String\_DisplayPos

### 37.5 MUIA\_String\_Contents – (V4) [ISG], STRPTR

#### FUNCTION

Get and set a string gadgets contents. You may not modify the returned string.

MUIA\_String\_Contents gets updated every time when the contents of the string gadget change. When you set up a notification on this attribute, you will hear about every keystroke.

**EXAMPLE**

```

/* The given hook will be called after every change */
/* in the string gadget. It receives a pointer to */
/* a pointer to the current contents in register a1 */
/* (see MUIM_CallHook for details) */

DoMethod(str,MUIM_Notify,
    MUIA_String_Contents, MUIV_EveryTime,
    str, 3, MUIM_CallHook, &hook, MUIV_TriggerValue);

```

**SEE ALSO**

MUIA\_String\_Accept, MUIA\_String\_Reject, MUIA\_String\_MaxLen

**37.6 MUIA\_String\_DisplayPos – (V4) [.SG], LONG****FUNCTION**

MUIA\_String\_DisplayPos can be used to get and set the number of the first character of the string to be displayed. This attribute is probably not very interesting.

**SEE ALSO**

MUIA\_String\_Contents, MUIA\_String\_BufferPos

**37.7 MUIA\_String\_EditHook – (V7) [ISG], struct Hook \*****FUNCTION**

When specified, MUI calls this hook as if it was a real string edit hook in a real string gadget. It receives a pointer to itself in A0, a pointer to a SGWork structure in A2 and a pointer to the message in A1.

The hook will be called before MUI's private edit hook, the result is unused.

**37.8 MUIA\_String\_Format – (V4) [I.G], LONG****SPECIAL INPUTS**

MUIV\_String\_Format\_Left  
 MUIV\_String\_Format\_Center  
 MUIV\_String\_Format\_Right

**FUNCTION**

Used to adjust the alignment of the input string.

**SEE ALSO**

MUIA\_String\_BufferPos, MUIA\_String\_DisPos, MUIA\_String\_Contents

### 37.9 MUIA\_String\_Integer – (V4) [ISG], ULONG

#### FUNCTION

Useful for turning a string gadget into an integer gadget. Setting this attribute puts the value with "%ld" into the gadget, getting it returns a longword containing the string gadgets contents as number.

You should set MUIA\_String\_Accept to "0123456789" or something like that to avoid wrong characters.

#### EXAMPLE

```
StringObject,
    MUIA_String_Accept , "0123456879",
    MUIA_String_Integer, 42,
End;
```

### 37.10 MUIA\_String\_MaxLen – (V4) [I.G], LONG

#### FUNCTION

Setup the maximum length for the string gadget. This attribute is only valid at object creation time.

Default maximum length is 80.

#### NOTE

The maximum length includes the 0-byte at the end of the string. To let the user enter e.g. 10 characters, you would have to specify a maxlen of 11.

#### SEE ALSO

MUIA\_String\_Contents

### 37.11 MUIA\_String\_Reject – (V4) [ISG], STRPTR

#### FUNCTION

A string containing characters that should not be accepted as input for the string gadget. Whenever the user hits such a char, he will hear a beep and gadgets contents won't have changed.

#### SEE ALSO

MUIA\_String\_Accept

### 37.12 MUIA\_String\_Secret – (V4) [I.G], BOOL

#### FUNCTION

This attribute causes the string gadget to display only dots instead of the real contents. Useful for password requesters.

#### SEE ALSO

MUIA\_String\_Contents



## 38 Text.mui

Text class allows generating objects that contain some kind of text. You can control the outfit of your text with some special control characters, including italics, bold, underline and color codes. Format codes align text either left, centered or right, linefeeds allow multiline text fields.

### 38.1 MUIA\_Text\_Contents – (V4) [ISG], STRPTR

#### FUNCTION

String to be displayed in a text object.

If the string is larger than available display space, it will be clipped. Setting MUIA\_Text\_Contents to NULL results in an empty text object.

The string is copied into a private buffer, you can destroy the original after using this tag.

Whenever MUI prints strings, they may contain some special character sequences defining format, color and style of the text.

**'\n'** Start a new line. With this character you can e.g. create multi line buttons.

**ESC -** Disable text engine, following chars will be printed without further parsing.

**ESC u** Set the soft style to underline.

**ESC b** Set the soft style to bold.

**ESC i** Set the soft style to italic.

**ESC n** Set the soft style back to normal.

**ESC <n>** Use pen number n (2..9) as front pen. n must be a valid DrawInfo pen as specified in "intuition/screens.h".

**ESC c** Center current (and following) line(s). This sequence is only valid at the beginning of a string or after a newline character.

**ESC r** Right justify current (and following) line(s). This sequence is only valid at the beginning of a string or after a newline character.

**ESC l** Left justify current (and following) line(s). This sequence is only valid at the beginning of a string or after a newline character.

**ESC I[s]** Draw MUI image with specification <s>. See autodocs of image class for image spec definition.

#### NOTE

These rules apply to all MUI strings, not only to a text objects contents. You can e.g. format the columns of a listview or include images in a cycle gadgets entries.

#### EXAMPLE

```
...
MUIA_Text_Contents, "\33c\33bMUI\33n\nis magic"
...
```

```
would look like      |      MUI      |  <-- bold
                      | is magic |  <-- normal
```

SEE\_ALSO

MUIA\_Text\_SetMin, MUIA\_Text\_SetMax, MUIA\_Text\_PreParse

### 38.2 MUIA\_Text\_HiChar – (V4) [I.], char

#### FUNCTION

If the character given here exists in the displayed string (no matter if upper or lower case), it will be underlined. This makes it easy to create macros such as KeyButton() that specify the control char and the underline char at the same time.

#### SEE ALSO

MUIA\_Text\_Contents, MUIA\_Control\_Char

### 38.3 MUIA\_Text\_PreParse – (V4) [ISG], STRPTR

#### FUNCTION

String containing format definitions to be parsed before the text from MUIA\_Text\_Contents is printed.

Using this tag, you can easily define different formats, colors and styles without modifying the original string.

#### EXAMPLE

```
...
MUIA_Text_PreParse, "\33c\33i",    // centered and italics
MUIA_Text_Contents, "foobar",
...
```

SEE\_ALSO

MUIA\_Text\_Contents

### 38.4 MUIA\_Text\_SetMax – (V4) [I.], BOOL

#### FUNCTION

Boolean value to indicate whether the objects maximal width shall be calculated to fit the string given with MUIA\_Text\_Contents.

When set to FALSE, maximum width is not limited.

For a text object that needs to be updated (e.g. some information about your programs status) you would probably set MUIA\_Text\_SetMax to FALSE to allow resizing of this object.

For a label for one of your gadgets, you might want to give this tag a value of TRUE to prevent MUI from inserting additional layout space.

Defaults to FALSE.

#### EXAMPLE

```
...
TX_Status = TextObject,
    RecessedFrame,
    MUIA_Background    , MUII_BACKGROUND,
    MUIA_Text_PreParse, "\33c",
```

```

    MUIA_Text_Contents, "running...",
    End,
...
set(TX_Status,MUIA_Text_Contents,"reading...");
...
set(TX_Status,MUIA_Text_Contents,"writing...");
...

SEE_ALSO
MUIA_Text_SetMin, MUIA_Text_Contents

```

### 38.5 MUIA\_Text\_SetMin – (V4) [I.], BOOL

#### FUNCTION

Boolean value to indicate whether the objects minimal width shall be calculated to fit the string given with MUIA\_Text\_Contents.

When set to FALSE, minimum width will be set to 0 and the displayed string may be clipped.

Defaults to TRUE.

SEE\_ALSO MUIA\_Text\_SetMax, MUIA\_Text\_Contents

## 39 Virtgroup.mui

Virtgroup class generates special kinds of group objects whose children can be a lot larger than the actual group. The group acts as a (small) window through which a rectangle area of its contents is visible.

Layout of a virtual groups children doesn't depend on the space available for the virtual group object. The children will get as much room as they want, usually their default size.

Virtual groups themselves don't offer any scrollbars to allow user interaction. These things are handled by scrollgroup class.

### 39.1 MUIA\_Virtgroup\_Height – (V6) [..G], LONG

#### FUNCTION

Read the virtual height of a virtual group.

#### NOTE

Currently you are unable to set the height, this might change in future releases.

#### SEE ALSO

MUIA\_Virtgroup\_Width, MUIA\_Virtgroup\_Left, MUIA\_Virtgroup\_Top

### 39.2 MUIA\_Virtgroup\_Left – (V6) [ISG], LONG

#### FUNCTION

Get/set the virtual left edge of a virtual group. The left edge will automatically be clipped to be between 0 and (VirtualWidth-DisplayWidth).

**SEE ALSO**

MUIA\_Virtgroup\_Width, MUIA\_Virtgroup\_Height, MUIA\_Virtgroup\_Top

**39.3 MUIA\_Virtgroup\_Top – (V6) [ISG], LONG****FUNCTION**

Get/set the virtual top edge of a virtual group. The top edge will automatically be clipped to be between 0 and (VirtualTop-DisplayTop).

**SEE ALSO**

MUIA\_Virtgroup\_Width, MUIA\_Virtgroup\_Height, MUIA\_Virtgroup\_Left

**39.4 MUIA\_Virtgroup\_Width – (V6) [..G], LONG****FUNCTION**

Read the virtual width of a virtual group.

**NOTE**

Currently you are unable to set the width, this might change in future releases.

**SEE ALSO**

MUIA\_Virtgroup\_Height, MUIA\_Virtgroup\_Left, MUIA\_Virtgroup\_Top

**40 Volumelist.mui**

Volumelist generates a list of all available volumes. Since you shouldn't use your own file requester in every application, this class is probably not of much use.

**41 Window.mui**

Objects of window class are used to generate windows and supply a place where MUI gadgets feel well. It handles the complicated task of window resizing fully automatic, you don't need to worry about that.

Windows are children of an application, you cannot use a window object without having a parent application object. On the other side, the gadgets in a window are children of the window, you cannot use MUI gadgets without having a parent MUI window.

Creating a window object does not mean to open it instantly. This is done later by setting the window's MUIA\_Window\_Open attribute. If your application has several windows, the usual way is to create them all at once at startup time and open/close it later just by setting MUIA\_Window\_Open.

There is no difference in talking to gadgets whether their parent window is open or not. If you e.g. set the contents of a string gadget in an open window, the gadget will refresh immediately. If the window is closed, the gadget just remembers its new setting and displays it later.

**41.1 MUIM\_Window\_GetMenuCheck (V4) (OBSOLETE)****SYNOPSIS**

DoMethod(obj,MUIM\_Window\_GetMenuCheck,ULONG MenuID);

**FUNCTION**

Ask whether a checkmark menu item has its checkmark set or cleared.

**INPUTS**

**MenuID** - the value you wrote into the UserData field of struct NewMenu.

**SEE ALSO**

MUIM\_Window\_SetMenuCheck, MUIA\_Window\_Menu

**41.2 MUIM\_Window\_GetMenuState (V4) (OBSOLETE)****SYNOPSIS**

DoMethod(obj,MUIM\_Window\_GetMenuState,ULONG MenuID);

**FUNCTION**

Ask whether a menu item is enabled or disabled.

**INPUTS**

**MenuID** - the value you wrote into the UserData field of struct NewMenu.

**SEE ALSO**

MUIM\_Window\_SetMenuState, MUIA\_Window\_Menu

**41.3 MUIM\_Window\_ScreenToBack (V4)****SYNOPSIS**

DoMethod(obj,MUIM\_Window\_ScreenToBack,);

**FUNCTION**

Put the window's screen to back. This command is only valid when the window is opened.

**SEE ALSO**

MUIM\_Window\_ScreenToFront, MUIM\_Window\_ToFront, MUIM\_Window\_ToBack

**41.4 MUIM\_Window\_ScreenToFront (V4)****SYNOPSIS**

DoMethod(obj,MUIM\_Window\_ScreenToFront,);

**FUNCTION**

Put the window's screen to front. This command is only valid when the window is opened.

**SEE ALSO**

MUIM\_Window\_ScreenToBack, MUIM\_Window\_ToFront, MUIM\_Window\_ToBack

**41.5 MUIM\_Window\_SetCycleChain (V4)****SYNOPSIS**

```
DoMethod(obj,MUIM_Window_SetCycleChain,Object *obj[1]);
```

**FUNCTION**

Set the cycle chain for a window. To make MUI's keyboard control work, you need to setup a chain of objects that should be activatable with the tab key. This can be any objects you wish, MUI supports complete keyboard handling even for sliders or listviews.

If you forget to set a cycle chain because you are a mouse-man, you certainly will annoy some users of your application!

**INPUTS**

One or more objects, terminated with a NULL.

**EXAMPLE**

```
DoMethod(window,MUIM_Window_SetCycleChain,
    str1,str2,slide1,list,radio,cycle1,cycle2,NULL);
```

**SEE ALSO**

MUIA\_Window\_ActiveObject

**41.6 MUIM\_Window\_SetMenuCheck (V4) (OBSOLETE)****SYNOPSIS**

```
DoMethod(obj,MUIM_Window_SetMenuCheck,ULONG MenuID, LONG stat);
```

**FUNCTION**

Set or clear the checkmark of a menu item.

**INPUTS**

**MenuID** - the value you wrote into the UserData field of struct NewMenu.

**set** - TRUE to set checkmark, FALSE to clear

**SEE ALSO**

MUIM\_Window\_GetMenuCheck, MUIA\_Window\_Menu,

## 41.7 MUIM\_Window\_SetMenuState (V4) (OBSOLETE)

### SYNOPSIS

DoMethod(obj,MUIM\_Window\_SetMenuState,ULONG MenuID, LONG stat);

### FUNCTION

Enable or disable a menu item.

### INPUTS

**MenuID** - the value you wrote into the UserData field of struct NewMenu.

**set** - TRUE to enable item, FALSE to disable.

### SEE ALSO

MUIM\_Window\_GetMenuState, MUIA\_Window\_Menu,

## 41.8 MUIM\_Window\_ToBack (V4)

### SYNOPSIS

DoMethod(obj,MUIM\_Window\_ToBack,);

### FUNCTION

Put the window to back. When the window is not currently open, this command does simply nothing.

### SEE ALSO

MUIM\_Window\_ToFront, MUIM\_Window\_ScreenToFront,  
MUIM\_Window\_ScreenToBack

## 41.9 MUIM\_Window\_ToFront (V4)

### SYNOPSIS

DoMethod(obj,MUIM\_Window\_ToFront,);

### FUNCTION

Put the window to front. When the window is not currently open, this command does simply nothing.

### SEE ALSO

MUIM\_Window\_ToBack, MUIM\_Window\_ScreenToFront,  
MUIM\_Window\_ScreenToBack

### 41.10 MUIA\_Window\_Activate – (V4) [ISG], BOOL

#### FUNCTION

Setting this to TRUE will activate the window. Setting this to FALSE has no effect. The attribute will change whenever the user activates/deactivates the window.

Specifying FALSE at object creation time will make the window open in an inactive state.

### 41.11 MUIA\_Window\_ActiveObject – (V4) [.SG], Object \*

#### SPECIAL INPUTS

MUIV\_Window\_ActiveObject\_None

MUIV\_Window\_ActiveObject\_Next

MUIV\_Window\_ActiveObject\_Prev

#### FUNCTION

Set the active object in a window as if the user would have activated it with the tab key. The object has to be in the cycle chain for this command to work.

#### EXAMPLE

```
set(window,MUIA_Window_ActiveObject,okaybutton);
```

#### SEE ALSO

MUIM\_Window\_SetCycleChain

### 41.12 MUIA\_Window\_AltHeight – (V4) [I.G], LONG

#### SPECIAL INPUTS

MUIV\_Window\_AltHeight\_MinMax(p)

MUIV\_Window\_AltHeight\_Visible(p)

MUIV\_Window\_AltHeight\_Screen(p)

MUIV\_Window\_AltHeight\_Scaled

#### FUNCTION

Specify the alternate (zoomed) height of a window. If not present, the alternate height will be the minimum height.

#### SEE ALSO

MUIA\_Window\_Height, MUIA\_Window\_AltWidth

### 41.13 MUIA\_Window\_AltLeftEdge – (V4) [I.G], LONG

#### SPECIAL INPUTS

MUIV\_Window\_AltLeftEdge\_Centered

MUIV\_Window\_AltLeftEdge\_Moused



MUIV\_Window\_AltLeftEdge\_NoChange

#### FUNCTION

Specify the alternate (zoomed) left position of a window. This defaults to the standard left position.

#### SEE ALSO

MUIA\_Window\_LeftEdge, MUIA\_Window\_AltTopEdge

### 41.14 MUIA\_Window\_AltTopEdge – (V4) [I.G], LONG

#### SPECIAL INPUTS

MUIV\_Window\_AltTopEdge\_Centered  
 MUIV\_Window\_AltTopEdge\_Moused  
 MUIV\_Window\_AltTopEdge\_Delta(p)  
 MUIV\_Window\_AltTopEdge\_NoChange

#### FUNCTION

Specify the alternate (zoomed) top position of a window. This defaults to the standard top position.

#### SEE ALSO

MUIA\_Window\_TopEdge, MUIA\_Window\_AltLeftEdge

### 41.15 MUIA\_Window\_AltWidth – (V4) [I.G], LONG

#### SPECIAL INPUTS

MUIV\_Window\_AltWidth\_MinMax(p)  
 MUIV\_Window\_AltWidth\_Visible(p)  
 MUIV\_Window\_AltWidth\_Screen(p)  
 MUIV\_Window\_AltWidth\_Scaled

#### FUNCTION

Specify the alternate (zoomed) width of a window. If not present, the alternate width will be the minimum width.

#### SEE ALSO

MUIA\_Window\_Width, MUIA\_Window\_AltHeight

### 41.16 MUIA\_Window\_AppWindow – (V5) [I.], BOOL

#### FUNCTION

Setting this attribute to TRUE will make this window an AppWindow, the user will be able to drop icons on it. You can hear about these events by listening to the MUIA\_AppMessage attribute.

**SEE ALSO**

MUIA\_AppMessage, MUIA\_Application\_DropObject

**41.17 MUIA\_Window\_Backdrop – (V4) [I.], BOOL****FUNCTION**

Make the window a backdrop window.

**41.18 MUIA\_Window\_Borderless – (V4) [I.], BOOL****FUNCTION**

Make the window borderless.

**41.19 MUIA\_Window\_CloseGadget – (V4) [I.], BOOL****FUNCTION**

Set this to FALSE and your window will not have a close gadget.

**41.20 MUIA\_Window\_CloseRequest – (V4) [..G], BOOL****FUNCTION**

When the user hits a windows close gadget, the window isn't closed immediately. Instead MUI only sets this attribute to TRUE to allow your application to react.

Usually, you will setup a notification that automatically closes the window when a close request appears, but you could e.g. pop up a confirmation requester or do some other things first.

**EXAMPLE**

```
/* automagically close a window      */
/* when the close gadget is pressed */

DoMethod(window,MUIM_Notify,
    MUIA_Window_CloseRequest, TRUE,
    window,3,MUIM_Set,MUIA_Window_Open,0);
```

**SEE ALSO**

MUIA\_Window\_Open

**41.21 MUIA\_Window\_DefaultObject – (V4) [ISG], Object \*****FUNCTION**

The default object in a window receives keyboard input as long as no other object is active. Good candidates for default objects are e.g. lonely listviews. Making such a listview the default object will allow the user to control it immediately without the need of several tab strokes for activation.

**SEE ALSO**

MUIA\_Window\_ActiveObject

**41.22 MUIA\_Window\_DepthGadget – (V4) [I.], BOOL****FUNCTION**

Enable or disable the depth gadget. Defaults to TRUE. There is no good reason to use this tag.

**41.23 MUIA\_Window\_DragBar – (V4) [I.], BOOL****FUNCTION**

Tell MUI to give your window a dragbar.

Defaults to TRUE.

There is no good reason to disable the dragbar!

**41.24 MUIA\_Window\_FancyDrawing – (V8) [ISG], BOOL****FUNCTION**

Usually, the only possible place to do some rendering is during a MUIM\_Draw method. However, if you have a class that really requires very high graphical output speed (e.g. a module players scope or a game class), you can set MUIA\_Window\_FancyDrawing to TRUE.

This allows your class to render anywhere between MUIM\_Show and MUIM\_Hide, e.g. directly after an attribute change with OM\_SET or from a separate task.

Note that your rastport etc. is only valid between MUIM\_Show and MUIM\_Hide. Keep that in mind!

When drawing from a separate task, you have to clone the RastPort and use the copy for your rendering!

**NOTE**

Please use this attribute sparingly. It might prevent MUI from doing nice things with your window, e.g. building an automatic virtual group when the screen is too small.

MUIA\_Window\_FancyDrawing is really only necessary for very few types of applications. You should use the traditional way (MUIM\_Draw and MUI\_Redraw()) whenever and wherever possible!

**41.25 MUIA\_Window\_Height – (V4) [I.G], LONG****SPECIAL INPUTS**

MUIV\_Window\_Height\_MinMax(p)

MUIV\_Window\_Height\_Visible(p)

MUIV\_Window\_Height\_Screen(p)

MUIV\_Window\_Height\_Scaled

MUIV\_Window\_Height\_Default

**FUNCTION**

Specify the height of a window. Usually, you won't give a pixel value here but instead use one of the following magic macros:

**MUIV\_Window\_Height\_Default:** calculated from objects default sizes.

**MUIV\_Window\_Height\_MinMax(0..100):** somewhere between the minimum height (0) and the maximum height (100) of your window.

**MUIV\_Window\_Height\_Visible(1..100):** percentage of the screens visible height.

**MUIV\_Window\_Height\_Screen(1..100):** percentage of the screens total height.

**MUIV\_Window\_Height\_Scaled:** height will be adjusted so that width : height == minimum width : minimum height. Note that a windows width and height may not both be scaled.

Default for this tag is MUIV\_Window\_Height\_Default.

As long as your window has a window id (MUIA\_Window\_ID), choosing a size is not that important. MUI will always remember a windows last position and size and these values will simply override your settings. Positioning and sizing should be completely under user control, a programmer doesn't need to worry about it.

#### SEE ALSO

MUIA\_Window\_Width, MUIA\_Window\_ID

### 41.26 MUIA\_Window\_ID – (V4) [ISG], ULONG

#### FUNCTION

For most of your windows, you should define a longword as id value. Only a window with an id is able to remember its size and position.

Additionally, when you use an ascii id (e.g. 'MAIN'), your window can be controlled from ARexx.

Of course all windows of your application must have unique ids.

#### SEE ALSO

MUIA\_Window\_LeftEdge

### 41.27 MUIA\_Window\_InputEvent – (V4) [..G], struct InputEvent \*

#### FUNCTION

This attribute gets set whenever your window receives a rawkey input event. You can react on this by creating a notification event containing a standard commodities.library input description string.

#### EXAMPLE

```
DoMethod(window, MUIM_Notify,
    MUIA_Window_InputEvent, "control p",
    txobj, 3,
    MUIM_Set, MUIA_Text_Contents, "user pressed ctrl p");
```

**41.28 MUIA\_Window\_LeftEdge – (V4) [I.G], LONG****SPECIAL INPUTS**

MUIV\_Window\_LeftEdge\_Centered

MUIV\_Window\_LeftEdge\_Moused

**FUNCTION**

Specify the left edge of a window. Usually, you shouldn't define a pixel value here but instead use one of the following macros:

**MUIV\_Window\_LeftEdge\_Centered:** window appears centered on the visible area of screen.

**MUIV\_Window\_LeftEdge\_Moused** window appears centered under the mouse pointer.

Default for this tag is MUIV\_Window\_LeftEdge\_Centered.

As long as your window has a window id (MUIA\_Window\_ID), choosing a position is not that important. MUI will always remember a windows last position and size and these values will simply override your settings. Positioning and sizing should be completely under user control, a programmer doesn't need to worry about it.

**SEE ALSO**

MUIA\_Window\_TopEdge, MUIA\_Window\_ID

**41.29 MUIA\_Window\_Menu – (V4) [I.], struct NewMenu \* (OBSOLETE)****SPECIAL INPUTS**

MUIV\_Window\_Menu\_NoMenu

**FUNCTION**

Obsolete, use MUIA\_Window\_Menustrip instead.

**SEE ALSO**

MUIA\_Window\_Menustrip

**41.30 MUIA\_Window\_MenuAction – (V8) [ISG], ULONG****FUNCTION**

Whenever a menu item is selected, this attribute will be set to the corresponding UserData field of the gadtools NewMenu structure. This allows reacting on menu items via broadcasting.

**SEE ALSO**

MUIA\_Window\_Menu

**41.31 MUIA\_Window\_Menustrip – (V8) [I.], Object \*****FUNCTION**

Specify a menu strip object for this window. The object is treated as a child of the window and will be disposed when the window is disposed.

Menustrip objects defined for a window will override an applications Menustrip object.

MUIA\_Window\_Menustrip replaces the old and obsolete MUIA\_Window\_Menu tag.

Usually, you will create the menu object with MUI's builtin object library from a gadtools NewMenu structure, but its also OK to define the menu tree "by hand" using the Family class.

If you have a global menu for all your applications windows but you want some windows to have no menu, use the MUIA\_Window\_NoMenus tag.

**SEE ALSO**

MUIA\_Window\_NoMenus

**41.32 MUIA\_Window\_MouseObject – (V10) [..G], Object \*****FUNCTION**

When MUIA\_Window\_NeedsMouseObject is enabled for this window, you can setup notificationns on MUIA\_Window\_MouseObject to find out on which object the mouse pointer is located.

**SEE ALSO**

MUIA\_Window\_NeedsMouseObject

**41.33 MUIA\_Window\_NeedsMouseObject – (V10) [I.], BOOL****FUNCTION**

If you want to react on changes of the MUIA\_Window\_MouseObject attribute, you have to set this to TRUE when creating your window.

**SEE ALSO**

MUIA\_Window\_MouseObject

**41.34 MUIA\_Window\_NoMenus – (V4) [IS.], BOOL****FUNCTION**

Temporarily disable the menu strip of a window.

**SEE ALSO**

MUIA\_Window\_Menu

**41.35 MUIA\_Window\_Open – (V4) [.SG], BOOL****FUNCTION**

This little attribute can be used to open and close a window. When opening a window, MUI does lots of stuff to calculate sizes and positions of all gadgets. Minimum and maximum window sizes will be adjusted automatically.

When the minimum size of a window is too big to fit on the screen, MUI tries to reduce font sizes and does a new calculation. You should always design your windows to fit on a 640\*200 screen with all fonts set to topaz/8.

When a window is closed (and you specified a MUIA\_Window\_ID), MUI remembers its position and size and uses these values during the next opening.

After setting MUIA\_Window\_Open to TRUE, you should test if MUI was able to open the window by getting the attribute again. If you don't and if this was the only window of your application, the user won't be able to do any input and your application will seem to hang.

**EXAMPLE**

```
set(window,MUIA_Window_Open,TRUE);
get(window,MUIA_Window_Open,&open);
if (!open)
{
    MUI_Request(app,0,0,0,"Ok","Failed to open window.");
    exit(20);
}
```

**SEE ALSO**

MUIA\_Window\_RootObject

**41.36 MUIA\_Window\_PublicScreen – (V6) [ISG], STRPTR****FUNCTION**

Force the window to appear on the public screen who's name is specified by this attribute. This tag overrides the user preferences setting and is overridden by MUIA\_Window\_Screen.

Please use this tag sparingly, overriding user prefs is not a good idea!

**SEE ALSO**

MUIA\_Window\_Screen

**41.37 MUIA\_Window\_RefWindow – (V4) [IS.], Object \*****FUNCTION**

Setting MUIA\_Window\_RefWindow to another MUI window object will make the left and top position relative to this reference window. Using the MUIA\_Window\_Left(Top)Edge-Centered tag, you can easily open one window within another.

Note that if your window has an id, the window will remember its last position and reopen there. Thus, this tag is only useful if you omit MUIA\_Window\_ID, maybe for some small requester windows.

**SEE ALSO**

MUIA\_Window\_ID, MUIA\_Window\_LeftEdge

**41.38 MUIA\_Window\_RootObject – (V4) [I.], Object \*****FUNCTION**

This is a pointer to a MUI object and defines the contents of your window. Usually, this root object will be of class MUIC\_Group since you surely want to have more than one gadget.

The root object is treated as child of a window and will be disposed when the window is disposed. Note that windows can only have one child.

Although you may create a window without root object, you have to set one before the window is opened!

**EXAMPLE**

```
win = WindowObject, MUIA_Window_RootObject,
    VGroup,
        Child, ...,
        Child, ...,
    End,
End;
```

**SEE ALSO**

MUIA\_Window\_Open

**41.39 MUIA\_Window\_Screen – (V4) [ISG], struct Screen \*****FUNCTION**

You can get a pointer to the parent screen of a window by getting this attribute. The result will be NULL when the window is currently closed.

Specifying MUIA\_Window\_Screen at object creation time or with a SetAttrs() call allows you to explicitly tell MUI on which screen the window should be opened. You normally won't need this feature and leave the decision about screens to the users preferences setting.

**SEE ALSO**

MUIA\_Window\_PublicScreen, MUIA\_Window\_Window

**41.40 MUIA\_Window\_ScreenTitle – (V5) [ISG], STRPTR****FUNCTION**

This text will appear in the screens title bar when the window is active.

**SEE ALSO**

MUIA\_Window\_Title



**41.41 MUIA\_Window\_SizeGadget – (V4) [I.], BOOL****FUNCTION**

Tell MUI if you want a sizing gadget for this window. Usually you won't need this attribute since MUI will automatically disable the sizing gadget when your window is not sizeable because of your gadget layout.

**41.42 MUIA\_Window\_SizeRight – (V4) [I.], BOOL****FUNCTION**

When set to TRUE, the size gadget will reside in the right window border.

**41.43 MUIA\_Window\_Sleep – (V4) [.SG], BOOL****FUNCTION**

This attribute can be used to put a window to sleep. The window gets disabled and a busy pointer appears.

The attribute contains a nesting count, if you tell your window to sleep twice, you will have to tell it to wake up twice too.

A sleeping window cannot be resized.

**SEE ALSO**

MUIA\_Application\_Sleep

**41.44 MUIA\_Window\_Title – (V4) [ISG], STRPTR****FUNCTION**

Specify the title of a window.

**SEE ALSO**

MUIA\_Window\_ScreenTitle

**41.45 MUIA\_Window\_TopEdge – (V4) [I.G], LONG****SPECIAL INPUTS**

MUIV\_Window\_TopEdge\_Centered

MUIV\_Window\_TopEdge\_Moused

MUIV\_Window\_TopEdge\_Delta(p)

**FUNCTION**

Specify the top edge of a window. Usually, you shouldn't define a pixel value here but instead use one of the following macros:

**MUIV\_Window\_TopEdge\_Centered:** window appears centered on the visible area of screen.

**MUIV\_Window\_TopEdge\_Moused** window appears centered under the mouse pointer.

**MUIV\_Window\_TopEdge\_Delta(p)** window appears p pixels below the screens title bar.

Default for this tag is `MUIV_Window_TopEdge_Centered`.

As long as your window has a window id (`MUIA_Window_ID`), choosing a position is not that important. MUI will always remember a windows last position and size and these values will simply override your settings. Positioning and sizing should be completely under user control, a programmer doesn't need to worry about it.

#### SEE ALSO

`MUIA_Window_LeftEdge`, `MUIA_Window_ID`

### 41.46 MUIA\_Window\_Width – (V4) [I.G], LONG

#### SPECIAL INPUTS

`MUIV_Window_Width_MinMax(p)`

`MUIV_Window_Width_Visible(p)`

`MUIV_Window_Width_Screen(p)`

`MUIV_Window_Width_Scaled`

`MUIV_Window_Width_Default`

#### FUNCTION

Specify the width of a window. Usually, you won't give a pixel value here but instead use one of the following magic macros:

**MUIV\_Window\_Width\_Default:** calculated from objects default sizes.

**MUIV\_Window\_Width\_MinMax(0..100):** somewhere between the minimum width (0) and the maximum width (100) of your window.

**MUIV\_Window\_Width\_Visible(1..100):** percentage of the screens visible width.

**MUIV\_Window\_Width\_Screen(1..100):** percentage of the screens total width.

**MUIV\_Window\_Width\_Scaled:** width will be adjusted so that width : height == minimum width : minimum height. Note that a windows width and height may not both be scaled.

Default for this tag is `MUIV_Window_Width_Default`.

As long as your window has a window id (`MUIA_Window_ID`), choosing a size is not that important. MUI will always remember a windows last position and size and these values will simply override your settings. Positioning and sizing should be completely under user control, a programmer doesn't need to worry about it.

#### SEE ALSO

`MUIA_Window_Height`, `MUIA_Window_ID`

### 41.47 MUIA\_Window\_Window – (V4) [..G], struct Window

\*

#### FUNCTION

When your window is open, you can obtain a pointer to the intuition Window structure with this tag and use it e.g. in an `asl.library` requester call.

Since the user can close your window any time (e.g. iconification), you must be prepared to receive a NULL pointer as result.

**SEE ALSO**

*MUIA\_Window\_Screen*