MUIRexx

COLLABORATORS			
	TITLE : MUIRexx		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 25, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

MUIRexx

1.1 MUIRexx.guide

```
MUIRexx
*******

   This is Edition 5 of the MUIRexx documentation,
30 March 1997, for MUIRexx Version 3.0.


Author: Russ Leighton <rleighto@violin.calpoly.edu>


  Update Information            Read this first.
  Introduction                  A brief encounter.
  Registration                  My plea for support.
  Installation                  Pretty simple actually.

  Command Reference             The mysteries revealed.

  Utilities                     A little help goes along way.

  Example Macro                 Just to peak your interest.

  Command-Port Options          Where the action's at.
  MUI Format Sequences          Making fancy text strings.
  MUI Image Specifications      How to get little MUI images.
  MUI List Format               How to get your lists to look just right.
  Attribute TAGs                Now to really complicate things...

  MagicUserInterface            The heart of MUIRexx.
  Acknowledgements              Could not have done it without you.

  History                       What has been done so far.

  Concept Index                 Look it up here.
  Command Index                 Where is that command anyway?
```

1.2 MUIRexx.guide/Update Information

```
Update Information
******************


  Version 3.0
  Version 2.2
  Version 2.1
  Version 2.0
```

1.3 MUIRexx.guide/Version 3.0

```
Version 3.0
===========

   This version of 'MUIRexx' is a major update from the previous
version. I decided to designate this release a major update since some
significant changes have been made that will require changes to
existing scripts.  I again apologize for any inconvenience this causes.
 One of the biggest changes is the addition of the callhook command
that provides more flexibility in defining gadget actions.  A
consequence of this new command is the removal of the PRESS, APP, DROP
options from gadget commands which will necessitate changes in existing
scripts.  Also, most LABEL options are now final arguments and as a
consequence labels will no longer require double quoting.

   The major changes and additions include:

   * Added callhook command

   * Removed PRESS,APP,DROP options from all other commands

   * Removed COMMAND option from group command

   * Removed TRIG VAL options from object command

   * switch objects can now have primary and alternate labels

   * Added REMOVE option to list object to allow removal of specific
     lines

   * Empty groups (and windows) no longer cause crashes

   * Groups can now be added dynamically to other groups

   * Added new group POP option for creation of popup groups

   * Added monitor command to open/close console for display of
     received command lines and error information

   * Added print command to output text to the console opened by the
     monitor command (useful for debugging purposes)
```

* Added capability to specify inline commands (i.e. ARexx string macros) for objects (see Command-Port Options)

* Literal strings (strings created by prepending with an '=') are no longer returned with the prepended equals (see list)

* Added SPEC option to button, text, and switch objects

* check and image object options changed (they are now specific instances of the text object)

1.4 MUIRexx.guide/Version 2.2

Version 2.2
==========

This release removes some bugs (see History) and adds a TITLE option for list objects. This much requested addition allows lists to have titles (a string that remains at the top of the list). Context sensitive menus can now be created by simply defining a menu inside a group definition. An option (SPEC) has been added to the popasl and poplist objects to allow specification of the popbutton object. Also, a parse routine has been written to replace the builtin parse routine in MUI. This effectivily allows for arbitrary length command lines (up to the limit imposed by 'ARexx').

Note that as of this release I am now making 'MUIRexx' "supportware", that is if you would like to show your support for further development (and also enhance the prospect of getting support) then please consider registering (see Registration).

1.5 MUIRexx.guide/Version 2.1

Version 2.1
==========

This release is a minor update to version 2.0 (see Version 2.0). Changes / additions include a more general 'view' command (HELP, NODE options as well as settable attributes), and new commands - 'aslrequest' that allows use of a standard ASL file requester within scripts and 'poplist' for creation of string gadgets with attached popup lists. Virtual groups are now supported (through the addition of the 'group' options SCROLL and VIRTUAL) The 'image' object has been changed and icon images (specified with the ICON option) now use an external MUI class (which is included and must be installed). Also, the 'list' command has been extended for better list handling. General improvements have been made including nonvolatile TAG string values (strings passed as TAG values are stored in allocated memory), string arrays for TAG values, and string formatting ('\n' for carriage

returns and '\xxx' for octal specification of characters).

Additionally, the previous documentation neglected to mention a feature of the 'list' object supporting multicolumn lists. Strings inserted into lists may consist of several strings separated with commas. If the 'list' is given a format (with the LIST_FORMAT attribute TAG) then these strings will be displayed in the appropriate column (as defined by the format).

1.6 MUIRexx.guide/Version 2.0

```
Version 2.0
===========
```

This version of 'MUIRexx' is a major update from the previous version. I decided to designate this release a major update since a lot has changed (see History) and most likely any scripts written for previous versions will break under this version. I apologize for any inconvenience this causes, but it was necessary to make major changes to implement some new capabilities. One of the biggest changes is that most options now require a keyword, so if you find that gadgets are missing (or even whole groups) then it is probably because a keyword is missing (like the LABEL keyword for example). Please have a look at the included example scripts to get an idea of how things should now look. Also, AmigaDOS 3.0 or better is now required because of the need for datatypes.library.

Some other major changes and additions include:

* Multiple labels specified in a comma delimited list (used to be specified as multiple arguments separated by space)

* Attribute TAGS can be set and retrieved, greatly improving flexibility and eliminating the need for some options (which have been removed, e.g. WEIGHT)

* New 'object' command for creation of generalized objects based on MUI internal and external classes as well as BOOPSI classes (also eliminated the need for some built-in objects, like 'scale')

* New 'method' command to allow creation of class methods which greatly enhances flexibility and eliminates need for some commands (which have been removed, e.g. 'muiset', 'config')

* Drag and drop operations are fully support for some objects

* Internal variables may be set or retrieved for passing data conveniently between 'ARexx' scripts

Note that if a script does fail then most likely the application will still be running (even if no window is open). In this case just issue the quit command to the application port.

1.7 MUIRexx.guide/Introduction

```
Introduction
************
```

'MUIRexx' is a program which serves as an interface between 'ARexx' (Copyright (C) 1987, William S. Hawes) and MagicUserInterface (Copyright (C) 1993-97, Stefan Stuntz). 'MUIRexx' does not provide complete access to all of the capabilities of MagicUserInterface (MUI), however, quite a lot of capability is implemented in 'MUIRexx' such as notification, icon buttons, application objects (objects that react to icons dropped on them), and drag/drop objects, as well as many standard MUI objects. Complete graphical user interfaces as well as full applications can be developed using 'MUIRexx' and 'ARexx' macros. Additionally, it is also possible to dynamically change or add objects after the application has been created.

Since MUI is an object oriented extension it was felt that the general flavor of object oriented programming (OOP) should be retained in the 'ARexx' implementation. Therefore, the command structure has a familiar OOP look to it which is somewhat of a departure from normal 'ARexx' programming construction.

1.8 MUIRexx.guide/Disclaimer

```
Disclaimer
==========
```

Basically, I am not in any way responsible if anything bad happens to you because of using this software. Say, for example, a mad MUI hater breaks into your house and smashes up your computer because you are using 'MUIRexx' then I am not responsible ;-)

Anyway, I hate disclaimers because the implication is that our society has become so pitiful that we feel we have to protect ourselves with legal disclaimers even though we, as freeware contributors, are providing a service for no charge. Therefore, I implore, if you are a person who is inclined to use/abuse the legal system in anyway that would cause harm to someone who is only trying to contribute to the community then please do not use this software.

1.9 MUIRexx.guide/Conditions

```
Conditions
==========
```

Actually, there aren't any concerning usage. I do not require anything from you to use this software, however, please consider registering 'MUIRexx' (see Registration). I have put quite a lot of effort into this so I would appreciate some feedback. If you discover any bugs tell me about them. If you have any ideas let me know. If you write any interesting applications send them my way. I can be reached by way of the following email address,

Russ Leighton <rleighto@violin.calpoly.edu>

or alternativily, my old email address is still valid (for now),

Russ Leighton <russ@sneezy.lancaster.ca.us>

Also, it should be understood that I retain the copyright to the software 'MUIRexx' and as such do not give permission to anyone to sell or claim this software as their own. I do not, however, make any such claim to the scripts provided or to any scripts written that make use of 'MUIRexx'. The full distribution may be freely distributed, provided that the original distribution remains intact. Also, no fee may be charged for the software except for any nominal media and/or shipping charge. Additionally, this software (including all distribution contents) may not be included on any commercial disks (including disk magazines and cover disks) without the author's expressed permission. Exceptions to this rule includes any disks/CDROMs distributed by Cronus (Fred Fish) or Aminet. A limited distribution of 'MUIRexx' (binary and readme file only) may be distributed with software, with prior consent, provided that I am given due credit and that the software package be provided to me at no cost.

1.10 MUIRexx.guide/Requirements

Requirements
============

The minimum system requirements needed to use 'MUIRexx' are as follows.

* An Amiga! (any should do)

* Version 3.0 of the Amiga operating system or higher

* MUI 3.0 or better (see MagicUserInterface)

* Icon.mcc custom class (included in the distribution)

* ARexx (running of course)

Additionally, to run the included macros the following may be needed.

* rexxsupport.library (needed by all the scripts)

* some default icons and tools (take a look at the 'deficon.rexx' script)

The following are nice-to-have but not absolutely necessary.

* A graphics card for high resolution and lots of colors

* NewIcons ((C) Nicola Salmoria/Philip A. Vedovatti/Eric Sauvageau) for 256 color icons (great for thumbnail previews of images)

* PictIcon ((C) Chad Randall) to generate those 256 color icons

1.11 MUIRexx.guide/Registration

Registration
****

As of version 2.2, 'MUIRexx' is now being distributed as "supportware". I am requesting that you consider registering 'MUIRexx' as a show of support for further development. Also, I will give registered users priority consideration for support and suggested changes/additions to 'MUIRexx'. I will continue to distribute updates periodically and will provide them without any crippled features, but I will offer more frequent updates to registered users.

To register please send the following information and cash, check, or money order of 15 US\$ to

Russell Leighton
1640 4th St.
Los Osos, CA USA
93402

Registration information:

Name: _____
Address: _____
(optional) _____
_____
_____
Email: _____

Computer: _____
Memory: _____
OS version: _____
MUI version:_____
Comments:

1.12 MUIRexx.guide/Installation

Installation
****

Just use the supplied install script. It will ask for a directory to install the demos and documentation. It will also prompt for the directory to place the 'MUIRexx' executable. Make sure the selected directory is in your global search path.

Also, the external MUI class, Icon.mcc, must be copied to MUI:Libs/MUI. I recommend using the supplied install script which will install the class and all supporting files.

'MUIRexx' may be started either from a CLI or from the WorkBench. The command line syntax for 'MUIRexx' is:

```
-> [run] MUIRexx (script) PORT (portname) HELP (guidename)
                          OUTPUT (outname) ERROUT (outname)
```

If (SCRIPT) is specified then the corresponding 'ARexx' script will be executed, otherwise, the code will wait until it receives a command, presumably from a subsequently executed 'ARexx' command or script. The PORT option will set the 'MUIRexx' host address to (PORTNAME). If not given then the host address will be MUIREXX. The HELP option specifies an AmigaGuide® file to be used for online help (see the NODE argument descriptions for selected gadgets). The OUTPUT and ERROUT options will turn on port monitoring which will result in a trace list of either all issued commands or just errors (and the commands associated with the errors) respectivily. The (OUTNAME) may be a file name or console device. If none is specified then the default is CON:.

To use 'MUIRexx' from the WorkBench a project icon must be supplied with the application startup script. The icon name should be the startup script name with the file extension .INFO replacing .REXX. Alternativily, a tooltype SCRIPT may be set to the name of the script to be executed on startup. The full path should be specified if the icon is not located in the same directory as the script. The default tool for the icon should be set to 'MUIRexx' (with the full path included if 'MUIRexx' is not in the global path). A tooltype PORT may be specified to set the host address name for the application. If this tooltype is not specified then the name will be set to the icon name (minus the .INFO extension). Also, a tooltype HELP may be included to specify the AmigaGuide® file as above. It is also recommended that the value for the stack size be set at a reasonable level (whatever that is). If problems occur running a script try setting the stack higher. Likewise, tooltypes can be defined for OUTPUT and ERROUT.

1.13 MUIRexx.guide/Command Reference

Command Reference
****

This chapter is included as a reference to the commands available for use in 'ARexx' macros.

Each command template is given in the standard DOS ReadArg form. That is each argument name is given separated by commas with flags specified (separated by a /). Within an 'ARexx' script command arguments should be separated by space (not commas). The flag definitions are as follows:

Flag	Definition
/K	Keyword required.
/A	Argument required.
/F	Final argument. The remainder of the line will be set to this argument.
/M	Multiple arguments (separated by space).
/S	Switch argument.

Note that some string arguments shown in the examples are surrounded by two sets of quotes (consisting of a pair of single quotes and a pair of double quotes). The general rule of thumb is if the argument is the final argument (indicated by a /F in the command template) then only a single pair of quotes is necessary. Otherwise, if the string contains spaces then the two sets of quotes are necessary. The reason for this is that 'ARexx' tries to interpret every token it encounters. If the token is surrounded by quotes then the token is interpreted by removing the quotes and leaving the enclosed string intact. Likewise, the DOS ReadArgs function (used by 'MUI' to parse incoming 'ARexx' command lines) parses arguments separated by spaces, therefore, strings with spaces must by enclosed by quotes (hence, the need for the two sets of quotes). An exception is if the argument is designated as the final argument in which case the ReadArgs function will set the remainder of the command line as the final argument.

Standard Commands	Provided with all MUI apps
Windows	The parent of all
Groups	The layout
Menus	What is on the menu today?
Objects	The children
Misc	Those commands that don't fit above

1.14 MUIRexx.guide/Standard Commands

Standard Commands
==========

These commands are standard with all MUI applications.

quit
hide
show
info

1.15 MUIRexx.guide/quit

```
-- Command: quit FORCE/S
    This command will end the 'MUIRexx' application, closing windows
    and freeing all associated memory.  Note that if a script fails
    'MUIRexx' may still actually be running.  Use this command to end
    the process by using an inline 'ARexx' command (e.g.  issue 'rx
    "address [portname] quit"' from a shell).
```

1.16 MUIRexx.guide/hide

```
-- Command: hide
    Hides (iconifies) the application.
```

1.17 MUIRexx.guide/show

```
-- Command: show
    Shows (pops up) an iconified application.
```

1.18 MUIRexx.guide/info

```
-- Command: info ITEM/A
    According to the given parameter the result string is filled with
    the following contents (or something reasonably close):

       * "title" - Name of the ARexx port

       * "author" - "Russell Leighton"

       * "copyright" - "Copyright ©1995-1997, Russell Leighton"

       * "description" - "MUI Rexx interface."

       * "version" - "$VER: MUIRexx 3.0"

       * "base" - Name of the ARexx port

       * "screen" - Name of the public screen
```

1.19 MUIRexx.guide/Windows

```
Windows
=======

   These commands are used for window creation and disposal. At least
one window is required.


  window                          Begin window definition
  endwindow                       End window definition
```

1.20 MUIRexx.guide/window

```
 -- Command: window ID/K, COMMAND/K, PORT/K, TITLE/K, CLOSE/S, ATTRS/K/M
     This command begins the definition of a window. All group and
     initial object definitions must be placed between a 'window' and
     'endwindow' pair.  The arguments are optional.

        * ID [I..] - an id can be assigned to a window for later
          reference. The id can be any combination of up to 5
          characters.

        * COMMAND [I..] - if given, a close gadget will be attached to
          the window.  If the null string is specified then the window
          will simply be disposed when the close gadget is selected,
          otherwise, the command string given will be executed.  For a
          complete description of this option see Command-Port Options.
           Note that the window title will be used to replace a '%s'
          contained in the command string.  Also, note that it is up to
          the programmer to insure that the window is in fact closed,
          presumably by the macro that is specified in this argument.

        * PORT [I..] - a specific host port may be specified by this
          argument. For a complete description of this option see
          Command-Port Options.

        * TITLE [I..] - the window may be given a title which will be
          displayed in the windows title bar.

        * CLOSE [.S.] - if this switch is given and an id specified (see
          ID argument) for an existing window then the window will be
          closed.

        * ATTRS [ISG] - with this option any MUIA attribute TAGs may be
          set or retrieved (see Attribute TAGs).

     Some useful TAGs for use with the window command are:

          TAG_Name =                      TAG_id          Flags Type

          Window_Activate =               0x80428d2f /* V4  isg BOOL */
          Window_AltHeight =              0x8042cce3 /* V4  i.g LONG */
```

```
        Window_AltLeftEdge =              0x80422d65 /* V4  i.g LONG */
        Window_AltTopEdge =               0x8042e99b /* V4  i.g LONG */
        Window_AltWidth =                 0x804260f4 /* V4  i.g LONG */
        Window_AppWindow =                0x804280cf /* V5  i.. BOOL */
        Window_Backdrop =                 0x8042c0bb /* V4  i.. BOOL */
        Window_Borderless =               0x80429b79 /* V4  i.. BOOL */
        Window_DepthGadget =              0x80421923 /* V4  i.. BOOL */
        Window_DragBar =                  0x8042045d /* V4  i.. BOOL */
        Window_Height =                   0x80425846 /* V4  i.g LONG */
        Window_IsSubWindow =              0x8042b5aa /* V4  isg BOOL */
        Window_LeftEdge =                 0x80426c65 /* V4  i.g LONG */
        Window_NoMenus =                  0x80429df5 /* V4  is. BOOL */
        Window_Open =                     0x80428aa0 /* V4  .sg BOOL */
        Window_PublicScreen =             0x804278e4 /* V6  isg STRPTR */
        Window_ScreenTitle =              0x804234b0 /* V5  isg STRPTR */
        Window_SizeGadget =               0x8042e33d /* V4  i.. BOOL */
        Window_SizeRight =                0x80424780 /* V4  i.. BOOL */
        Window_Sleep =                    0x8042e7db /* V4  .sg BOOL */
        Window_TopEdge =                  0x80427c66 /* V4  i.g LONG */
        Window_UseBottomBorderScroller =  0x80424e79 /* V13 is. BOOL */
        Window_UseLeftBorderScroller =    0x8042433e /* V13 is. BOOL */
        Window_UseRightBorderScroller =   0x8042c05e /* V13 is. BOOL */
        Window_Width =                    0x8042dcae /* V4  i.g LONG */
        InnerBottom =                     0x8042f2c0 /* V4  i.g LONG */
        InnerLeft =                       0x804228f8 /* V4  i.g LONG */
        InnerRight =                      0x804297ff /* V4  i.g LONG */
        InnerTop =                        0x80421eb6 /* V4  i.g LONG */

        Note: Consult MUI:Developer/Autodocs/MUI_Window.doc and
              MUI:Developer/C/Include/libraries/mui.h

    Example use of this command:

        window ID DOCK ATTRS InnerBottom 0 InnerLeft 0 InnerRight 0 InnerTop 0
            .
            .
            .
        endwindow

        window ID DOCK ATTRS Window_Open
        say result
```

1.21 MUIRexx.guide/endwindow

```
-- Command: endwindow
    This command ends the definition of a window. All group and initial
    object definitions must be placed between a 'window' and
    'endwindow' pair. The defined window will be opened after issuing
    this command.
```

1.22 MUIRexx.guide/Groups

```
Groups
======
```

The following commands are used for group creation. Some arguments can be used after the groups window is opened to change or retrieve settings. In the argument descriptions the keywords are followed by a pair of [] containing indicator letters. If the letter I is indicated then the argument is valid during creation. If the letter S is indicated then the argument can be set after the groups window is open. If the letter G is indicated then the argument can be retrieved.

group
endgroup

1.23 MUIRexx.guide/group

-- Command: group ID/K, HELP/K, NODE/K, FRAME/S, HORIZ/S, REGISTER/S, VIRTUAL/S, SCROLL/S, POP/S, ICON/K, SPEC/K, ATTRS/K/M, LABEL=LABELS/K/F

This command begins the definition of a group. Groups are defined by placement of other groups and objects between a 'group' and 'endgroup' pair. The arguments are optional.

* ID [I..] - an id can be assigned to a group for later reference. The id can be any combination of up to 5 characters. If the id is given without any other arguments, and the group has been previously created then the group will be placed into a temporary state where objects can be added or existing objects changed. The 'endgroup' command will terminate this temporary state and cause the affected window to be updated. In this manner a window's contents may be dynamically altered. In particular, object settings that can only be specified when they are initially created (those arguments indicated with a I) can be changed utilizing this feature.

* HELP [I..] - with this argument help text may be defined which will be displayed as balloon help whenever the pointer is over the associated group. Of course, this is dependant on whether the user set up balloon help in the MUI preference settings.

* NODE [I..] - this argument is used to specify a node in the guide file given in the command line argument HELP for 'MUIRexx'. If the user positions the mouse pointer over the group and presses the help button on the keyboard then the guide file will be displayed at the node location.

* FRAME [I..] - if this switch is given then a frame will be rendered for the group.

* HORIZ [I..] - if this switch is given then the group will be arranged horizontally. If not specified then the group will be arranged vertically.

* REGISTER [I..] - if this switch is given then the group will be defined as a register group (i.e. a group consisting of pages of objects or groups). The LABELS argument must be given if this switch is specified.

* VIRTUAL [I..] - if this switch is given then the group will be defined as a virtual group (i.e. a group whose contents can be larger than the display region). This option will result in a virtual group with no scroll bars. The virtual group can be scrolled, however, by using the mouse (i.e. press and hold down the left mouse button while in the group and move past the edges or if you have a middle mouse button press and hold it while in the group and just move the mouse).

* SCROLL [I..] - if this switch is given then the group will be defined as a virtual group (i.e. a group whose contents can be larger than the display region). This option will result in a virtual group with scroll bars. The virtual group can be scrolled with the scroll bar gadgets or alternativily with the mouse as described above.

* POP [I..] - if this switch is given then the group will be created as a popup group (i.e. the group will be displayed if the associated popup object is pressed). A pop button image may be specified using either the ICON or SPEC or LABEL options, otherwise, a default image ('MUII_PopUp') will be used. Note that the contents of popup groups cannot be changed (by using the temporary change state as described above).

* ICON [I..] - the name of an icon may be specified with this argument. If given then the popup image for a pop group (if POP was given) will be set to the icon image. Note that the name of the icon should not be specified with a ".info".

* SPEC [I..] - this argument is used to specify a MUI image specification (see MUI Image Specifications) for a pop group (if POP was given).

* LABEL=LABELS [ISG] - either a group title, register labels, or popup label are specified with this option. Multiple labels are separated by commas. If the REGISTER switch was given then the labels correspond to the page titles. For each label specified there must be a corresponding group or object defined. If the REGISTER switch was not given then the first label will be rendered as a group title. For register groups, if the group was previously created (and its window is open) then the current page may be set by issuing the group command, with an existing ID, and a label corresponding to the page to be activated. Also, if an ID and the REGISTER switch are given without a label then the currently displayed page label will be returned in the 'ARexx'

```
        variable RESULT (if 'options results' was specified in the
        'ARexx' script).

      * ATTRS [ISG] - with this option any MUIA attribute TAGs may be
        set or retrieved (see Attribute TAGs).

    Some useful TAGs for use with the group command are:

        TAG_Name =                          TAG_id          Flags Type

        Group_ActivePage =                  0x80424199 /* V5  isg LONG */
        Group_Columns =                     0x8042f416 /* V4  is. LONG */
        Group_Horiz =                       0x8042536b /* V4  i.. BOOL */
        Group_HorizSpacing =                0x8042c651 /* V4  isg LONG */
        Group_PageMode =                    0x80421a5f /* V5  i.. BOOL */
        Group_Rows =                        0x8042b68f /* V4  is. LONG */
        Group_SameHeight =                  0x8042037e /* V4  i.. BOOL */
        Group_SameSize =                    0x80420860 /* V4  i.. BOOL */
        Group_SameWidth =                   0x8042b3ec /* V4  i.. BOOL */
        Group_Spacing =                     0x8042866d /* V4  is. LONG */
        Group_VertSpacing =                 0x8042e1bf /* V4  isg LONG */

        Note: Consult MUI:Developer/Autodocs/MUI_Group.doc and
              MUI:Developer/C/Include/libraries/mui.h

    Example use of this command:

        window ID MDIR
            group ID REG REGISTER LABELS 'Directory,Buffers,Volumes,Mirror'
            .
            .
            .
        endwindow

        group ID REG REGISTER
        say result
```

1.24 MUIRexx.guide/endgroup

```
 -- Command: endgroup
     This command ends the definition of a group. Groups are defined by
     placement of other groups and objects between a 'group' and
     'endgroup' pair.
```

1.25 MUIRexx.guide/Menus

```
Menus
=====
```

The following commands are used for menu creation.

menu
endmenu
item

1.26 MUIRexx.guide/menu

-- Command: menu ID/K, ATTRS/K/M, LABEL/K/F
This command begins the definition of a menu. Menus are defined by placement of other menus and menu items between a 'menu' and 'endmenu' pair. If a menu is defined within a window definition (i.e. between a WINDOW and ENDWINDOW pair), but outside any group definitions then the menu will be added to the window menu strip. If a menu is defined within a group then a context sensitive menu will be created for the group (i.e. when the mouse pointer is over the group and the right mouse button is pressed then a popup menu will be displayed). Note that only one menu may be defined for a context sensitive menu (all others will be ignored).

* ID [I..] - an id can be assigned to a menu for later reference. The id can be any combination of up to 5 characters.

* LABEL [I..] - the menu must be given a label which will be displayed in the menu bar of the window.

* ATTRS [ISG] - with this option any MUIA attribute TAGs may be set or retrieved (see Attribute TAGs).

Some useful TAGs for use with the menu command are:

```
TAG_Name =                          TAG_id          Flags Type

Menu_Enabled =                      0x8042ed48 /* V8  isg BOOL */
```

Note: Consult MUI:Developer/Autodocs/MUI_Menu.doc, MUI_Area.doc and MUI:Developer/C/Include/libraries/mui.h

1.27 MUIRexx.guide/endmenu

-- Command: endmenu
This command ends the definition of a menu. Menus are defined by placement of other menus and menu items between a 'menu' and 'endmenu' pair.

1.28 MUIRexx.guide/item

```
-- Command: item ID/K, COMMAND/K, PORT/K, ATTRS/K/M, LABEL/K/F
    This command defines a menu item.  This command is only valid
    between a 'menu' and 'endmenu' pair.

       * ID [I..] - an id can be assigned to a menu item for later
         reference. The id can be any combination of up to 5
         characters.

       * COMMAND [I..] - if given, the command will be executed
         whenever the menu item is selected.  For a complete
         description of this option see Command-Port Options.  Note
         that the menu item label (see the LABEL argument description)
         will be used to replace a '%s' contained in the command
         string.

       * PORT [I..] - a specific host port may be specified by this
         argument. For a complete description of this option see
         Command-Port Options.

       * LABEL [I..] - this is the text for the menu item (which must
         be given).

       * ATTRS [ISG] - with this option any MUIA attribute TAGs may be
         set or retrieved (see Attribute TAGs).

    Some useful TAGs for use with the item command are:

         TAG_Name =                          TAG_id          Flags Type

         Menuitem_Checked =                  0x8042562a /* V8  isg BOOL */
         Menuitem_Checkit =                  0x80425ace /* V8  isg BOOL */
         Menuitem_Enabled =                  0x8042ae0f /* V8  isg BOOL */
         Menuitem_Exclude =                  0x80420bc6 /* V8  isg LONG */
         Menuitem_Shortcut =                 0x80422030 /* V8  isg STRPTR */
         Menuitem_Title =                    0x804218be /* V8  isg STRPTR */
         Menuitem_Toggle =                   0x80424d5c /* V8  isg BOOL */

         Note: Consult MUI:Developer/Autodocs/MUI_Menuitem.doc and
               MUI:Developer/C/Include/libraries/mui.h

    Example use of this command:

         window TITLE '"MUIRexx Demo"' COMMAND '"quit"' PORT DEMO
             menu LABEL '"Project"'
                  .
                  .
                  .
                  item ATTRS Menuitem_Title '-1'  /* item separator bar */
                  item COMMAND '"quit"' PORT DEMO LABEL 'Quit'
             endmenu
             .
             .
             .
         endwindow
```

1.29 MUIRexx.guide/Objects

```
Objects
=======

   The following commands are used to create and manipulate objects.
Some arguments can be used after the object's window is opened to
change or retrieve settings.  Also, issuing an object command with only
an ID argument will return a result in the 'ARexx' variable RESULT (if
'options results' was specified in the 'ARexx' script).  In the
argument descriptions the keywords are followed by a pair of []
containing indicator letters.  If the letter I is indicated then the
argument is valid during creation.  If the letter S is indicated then
the argument can be set after the objects window is open.  If the letter
G is indicated then the argument can be retrieved.


  space
  label
  view
  gauge
  meter
  text
  button
  switch
  image
  check
  cycle
  radio
  string
  popasl
  poplist
  slider
  popslider
  knob
  list
  dirlist
  volumelist
  object
```

1.30 MUIRexx.guide/space

```
 -- Command: space BAR/S, HORIZ/S, VALUE
     This object will place some white space into the current location.
     A consequence of placing space is that the window will be sizable
     in the associated direction. The arguments are optional.

        * BAR [I..] - if this switch is given then a bar will be
```

```
      rendered in the center of the space.

    * HORIZ [I..] - if this switch is given then the space will be a
      horizontal space (best used within a horizontal group)
      otherwise the space will be vertical.

    * VALUE [I..] - this argument, if given, specifies the minimum
      amount of space.  The default value is zero.

Example use of this command:

      window
          .
          .
          .
          group HORIZ
              button LABEL 'OK'
              space BAR 0
              button LABEL 'Cancel'
          endgroup
      endwindow
```

1.31 MUIRexx.guide/label

```
-- Command: label LEFT/S, CENTER/S, SINGLE/S, DOUBLE/S, LABEL/A/F
    This object is a simple label that can be used in conjunction with
    another object for clarification of the other objects purpose.

    * LEFT [I..] - this switch will force the label to be left
      justified.

    * CENTER [I..] - this switch will force the label to be
      centered.

    * SINGLE [I..] - this switch will cause extra vertical space to
      be added to the label to center it about the same space
      occupied by an object with a single width frame.

    * DOUBLE [I..] - this switch will cause extra vertical space to
      be added to the label to center it about the same space
      occupied by an object with a double width frame.

    * LABEL [I..] - this is the text for the label (which must be
      the last argument and may contain spaces).

Example use of this command:

      window
          .
          .
          .
          group HORIZ
              label '"string:"'
```

```
            string ID STR1
        endgroup
    endwindow
```

1.32 MUIRexx.guide/view

```
-- Command: view ID/K, HELP/K, NODE/K, FILE/K, ATTRS/K/M, STRING/K/F
    This specifies a view object.

       * ID [I..] - an id can be assigned to a view for later
         reference. The id can be any combination of up to 5
         characters.

       * HELP [I..] - with this argument help text may be defined
         which will be displayed as balloon help whenever the pointer
         is over the associated view object.  Of course, this is
         dependant on whether the user set up balloon help in the MUI
         preference settings.

       * NODE [I..] - this argument is used to specify a node in the
         guide file given in the command line argument HELP for
         'MUIRexx'.  If the user positions the mouse pointer over the
         view object and presses the help button on the keyboard then
         the guide file will be displayed at the node location.

       * FILE [I..] - If this argument is given then the view contents
         will be retrieved from the specified file.

       * STRING [I..] - this is the view content string.  Note that
         the string may contain any of the special formatting
         sequences (see MUI Format Sequences).  Also, if this argument
         is given then it must be the last specified.

       * ATTRS [ISG] - with this option any MUIA attribute TAGs may be
         set or retrieved (see Attribute TAGs).

    Some useful TAGs for use with the gauge command are:

         TAG_Name =                          TAG_id          Flags Type

         Floattext_Justify =                 0x8042dc03 /* V4  isg BOOL */
         Floattext_SkipChars =               0x80425c7d /* V4  is. STRPTR */
         Floattext_TabSize =                 0x80427d17 /* V4  is. LONG */
         Floattext_Text =                    0x8042d16a /* V4  isg STRPTR */

         Note: Consult MUI:Developer/Autodocs/MUI_Floattext.doc and
               MUI:Developer/C/Include/libraries/mui.h

    Example use of this command:

         window ID LHA TITLE '"Archive List"'
             view ID LVIEW FILE '"ram:t/lha.out"'
             .
```

```
        .
        .
    endwindow

    view ID LVIEW ATTRS Floattext_Text
    say import(d2c(result))
```

1.33 MUIRexx.guide/gauge

```
-- Command: gauge ID/K, HELP/K, NODE/K, ATTRS/K/M, LABEL/K/F
    Gauge objects are created with this command.

       * ID [I..] - an id can be assigned to a gauge for later
         reference. The id can be any combination of up to 5
         characters.

       * HELP [I..] - with this argument help text may be defined
         which will be displayed as balloon help whenever the pointer
         is over the associated gauge. Of course, this is dependant on
         whether the user set up balloon help in the MUI preference
         settings.

       * NODE [I..] - this argument is used to specify a node in the
         guide file given in the command line argument HELP for
         'MUIRexx'. If the user positions the mouse pointer over the
         gauge and presses the help button on the keyboard then the
         guide file will be displayed at the node location.

       * LABEL [I..] - if given this label will be displayed in the
         gauge. A format specifier '%ld' may be included in the string
         which will be replaced by the current gauge level.

       * ATTRS [ISG] - with this option any MUIA attribute TAGs may be
         set or retrieved (see Attribute TAGs).

    Some useful TAGs for use with the gauge command are:

         TAG_Name =                          TAG_id          Flags Type

         Gauge_Current =                     0x8042f0dd /* V4  isg LONG */
         Gauge_Divide =                      0x8042d8df /* V4  isg BOOL */
         Gauge_Horiz =                       0x804232dd /* V4  i.. BOOL */
         Gauge_InfoText =                    0x8042bf15 /* V7  isg STRPTR */
         Gauge_Max =                         0x8042bcdb /* V4  isg LONG */

         Note: Consult MUI:Developer/Autodocs/MUI_Gauge.doc and
               MUI:Developer/C/Include/libraries/mui.h

    Example use of this command:

         window TITLE '"Test"' COMMAND '"quit"' PORT DEMO
             .
             .
```

```
        .
        group
            slider ID SLDR
            gauge ID GAUG LABEL '"level %ld"' ATTRS Gauge_Horiz TRUE
            object CLASS '"Scale.mui"'
            .
            .
            .
            group HORIZ
                group
                    label DOUBLE '"Hue:"'
                    label DOUBLE '"Saturation:"'
                endgroup
                group
                    gauge ID HUE ATTRS Gauge_Max 16384,
                                       Gauge_Divide 262144,
                                       Gauge_Horiz TRUE
                    gauge ID SAT ATTRS Gauge_Max 16384,
                                       Gauge_Divide 262144,
                                       Gauge_Horiz TRUE
                endgroup
            endgroup
        endgroup
    endwindow

    method ID SLDR Notify Numeric_Value EveryTime,
                   @GAUG 3 Set Gauge_Current TriggerValue
```

1.34 MUIRexx.guide/meter

```
-- Command: meter ID/K, HELP/K, NODE/K, ATTRS/K/M, LABEL/K/F
    Meter objects are created with this command.

      * ID [I..] - an id can be assigned to a meter for later
        reference. The id can be any combination of up to 5
        characters.

      * HELP [I..] - with this argument help text may be defined
        which will be displayed as balloon help whenever the pointer
        is over the associated meter. Of course, this is dependant on
        whether the user set up balloon help in the MUI preference
        settings.

      * NODE [I..] - this argument is used to specify a node in the
        guide file given in the command line argument HELP for
        'MUIRexx'. If the user positions the mouse pointer over the
        meter and presses the help button on the keyboard then the
        guide file will be displayed at the node location.

      * LABEL [I..] - if given this label will be displayed in the
        meter.

      * ATTRS [ISG] - with this option any MUIA attribute TAGs may be
```

```
     set or retrieved (see Attribute TAGs).

Some useful TAGs for use with the meter command are:

     TAG_Name =                          TAG_id          Flags Type

     Numeric_Default =                   0x804263e8 /* V11 isg LONG */
     Numeric_Format =                    0x804263e9 /* V11 isg STRPTR */
     Numeric_Max =                       0x8042d78a /* V11 isg LONG */
     Numeric_Min =                       0x8042e404 /* V11 isg LONG */
     Numeric_Reverse =                   0x8042f2a0 /* V11 isg BOOL */
     Numeric_RevLeftRight =              0x804294a7 /* V11 isg BOOL */
     Numeric_RevUpDown =                 0x804252dd /* V11 isg BOOL */
     Numeric_Value =                     0x8042ae3a /* V11 isg LONG */

     Note: Consult MUI:Developer/Autodocs/MUI_Numeric.doc and
           MUI:Developer/C/Include/libraries/mui.h

Example use of this command:

     window TITLE '"Test"' COMMAND '"quit"' PORT DEMO
         .
         .
         .
         group
             group HORIZ
                 knob ID KNOB HELP '"an example knob gadget"' NODE '"knob"'
                 meter ID METR NODE '"meter"' LABEL 'meter'
             endgroup
         endgroup
     endwindow

     method ID KNOB Notify Numeric_Value EveryTime,
                    @METR 3 Set Numeric_Value TriggerValue
```

1.35 MUIRexx.guide/text

```
-- Command: text ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, ICON/K,
         SPEC/K, PICT/K, TRANS/S, ATTRS/K/M, LABEL/K/F
    Text gadget objects are created with this command.  Text, button,
    switch, check, and image gadgets are essentially identical with
    the only differences being the base object class and the default
    appearance/action. All options are identical for these objects.

       * ID [I..] - an id can be assigned to a gadget for later
         reference. The id can be any combination of up to 5
         characters.  If the id is given without any other arguments,
         and the object has been previously created, then the label
         will be returned in RESULT (if OPTIONS RESULTS is specified
         in the script).

       * COMMAND [I..] - if given, the command will be executed
         whenever the gadget is pressed.  For a complete description
```

of this option see Command-Port Options. Note that the gadget label (see the LABEL argument description) will be used to replace a ‘%s’ contained in the command string. Additionally, if the gadget is an icon (specified by the ICON argument) and a command is not specified but a port is (see PORT argument description) then the command will be set to the default tool of the icon. For example, if the icons default tool is ‘MultiView’ then the command string will be set to ‘MultiView %s’.

* PORT [I..] - a specific host port may be specified by this argument. For a complete description of this option see Command-Port Options.

* HELP [I..] - with this argument help text may be defined which will be displayed as balloon help whenever the pointer is over the associated gadget. Of course, this is dependant on whether the user set up balloon help in the MUI preference settings.

* NODE [I..] - this argument is used to specify a node in the guide file given in the command line argument HELP for ‘MUIRexx’. If the user positions the mouse pointer over the gadget and presses the help button on the keyboard then the guide file will be displayed at the node location.

* ICON [I..] - the name of an icon may be specified with this argument. If given then the gadget image will be set to the icon image. Note that the name of the icon should not be specified with a ".info".

* SPEC [I..] - this argument is used to specify a MUI image specification (see MUI Image Specifications). For ‘check’ objects if not given then the image ‘MUII_CheckMark’ will be used.

* PICT [I..] - the name of a picture may be specified with this argument. If given then the gadget image will be set to the picture content. Any picture with an associated installed datatype may be used.

* TRANS [I..] - if this flag is given then the background color of the picture (defined with the PICT option) will be transparent.

* LABEL=LABELS [ISG] - the label for the gadget is given by this argument. Note that the string may contain any of the special formatting sequences (see MUI Format Sequences). Additionally, even though the label is not displayed for image gadgets it may still be used in the command string (see COMMAND argument above). Also, for ‘switch’ and ‘check’ objects two strings may be given (separated by a comma) that will be returned depending on the select state of the gadget. The first defines the unselected string and the second defines the selected. If not specified then the unselected string will be set to 0 and the selected string will be set to 1.

```
   * ATTRS [ISG] - with this option any MUIA attribute TAGs may be
     set or retrieved (see Attribute TAGs).

Some useful TAGs for use with this command are:

     TAG_Name =                          TAG_id          Flags Type

     Text_Contents =                     0x8042f8dc /* V4  isg STRPTR */
     Text_HiChar =                       0x804218ff /* V4  i.. char */
     Text_PreParse =                     0x8042566d /* V4  isg STRPTR */
     Text_SetMax =                       0x80424d0a /* V4  i.. BOOL */
     Text_SetMin =                       0x80424e10 /* V4  i.. BOOL */
     Text_SetVMax =                      0x80420d8b /* V11 i.. BOOL */
     ControlChar =                       0x8042120b /* V4  isg char */
     CycleChain =                        0x80421ce7 /* V11 isg LONG */
     Disabled =                          0x80423661 /* V4  isg BOOL */
     Draggable =                         0x80420b6e /* V11 isg BOOL */
     FixHeight =                         0x8042a92b /* V4  i.. LONG */
     FixHeightTxt =                      0x804276f2 /* V4  i.. STRPTR */
     FixWidth =                          0x8042a3f1 /* V4  i.. LONG */
     FixWidthTxt =                       0x8042d044 /* V4  i.. STRPTR */
     HorizDisappear =                    0x80429615 /* V11 isg LONG */
     HorizWeight =                       0x80426db9 /* V4  isg WORD */
     MaxHeight =                         0x804293e4 /* V11 i.. LONG */
     MaxWidth =                          0x8042f112 /* V11 i.. LONG */
     Selected =                          0x8042654b /* V4  isg BOOL */
     ShowMe =                            0x80429ba8 /* V4  isg BOOL */
     ShowSelState =                      0x8042caac /* V4  i.. BOOL */
     VertDisappear =                     0x8042d12f /* V11 isg LONG */
     VertWeight =                        0x804298d0 /* V4  isg WORD */
     Weight =                            0x80421d1f /* V4  i.. WORD */

     Note: Consult MUI:Developer/Autodocs/MUI_Text.doc, MUI_Image.doc,
           MUI_Area.doc and MUI:Developer/C/Include/libraries/mui.h

Example use of this command:

     window ID MRX1 TITLE '"demo"' COMMAND '"window ID MRX1 CLOSE"' PORT DEMO
         text LABEL 'A demonstration of MUIRexx'
         button ID BUT COMMAND '"out %s"' HELP '"button 1"' LABEL 'button 1'
         switch ID SWCH LABEL 'switch'
         check ID CHK1 STRINGS '"no,yes"' ATTRS Selected TRUE
         image SPEC '"4:MUI:Images/WD/13pt/PopUp.mf0"'
         .
         .
         .
     endwindow

     switch ID SWCH ATTRS Selected
     say result
```

1.36 MUIRexx.guide/button

```
-- Command: button ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, ICON/K,
         SPEC/K, PICT/K, TRANS/S, ATTRS/K/M, LABEL/K/F
    Button gadget objects are created with this command.  Text,
    button, switch, check, and image gadgets are essentially identical
    with the only differences being the base object class and the
    default appearance/action. All options are identical for these
    objects.  Refer to the text command for descriptions of the
    options.
```

1.37 MUIRexx.guide/switch

```
-- Command: switch ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, ICON/K,
         SPEC/K, PICT/K, TRANS/S, ATTRS/K/M, LABEL=LABELS/K/F
    Switch gadget objects are created with this command.  Text,
    button, switch, check, and image gadgets are essentially identical
    with the only differences being the base object class and the
    default appearance/action. The switch gadget differs from text
    gadgets in that its select state toggles with each press.
    Additionally, two labels may be given (using the LABEL option)
    seperated by a comma.  The first label will be displayed when the
    switch is unselected and the second when the switch is selected.
    All options are identical for these objects.  Refer to the text
    command for descriptions of the options.
```

1.38 MUIRexx.guide/image

```
-- Command: image ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, ICON/K,
         SPEC/K, PICT/K, TRANS/S, ATTRS/K/M, LABEL/K/F
    Image gadget objects are created with this command.  Text, button,
    switch, check, and image gadgets are essentially identical with
    the only differences being the base object class and the default
    appearance/action (image objects, by default, use MUI images
    specified through the SPEC option and do not toggle their state).
    All options are identical for these objects.  Refer to the text
    command for descriptions of the options.
```

1.39 MUIRexx.guide/check

```
-- Command: check ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, ICON/K,
         SPEC/K, PICT/K, TRANS/S, ATTRS/K/M, LABEL=LABELS/K/F
    Check gadget objects are created with this command.  Text, button,
```

switch, check, and image gadgets are essentially identical with the only differences being the base object class and the default appearance/action (check objects, by default, use MUI images specified through the SPEC option and toggle their state). All options are identical for these objects. Refer to the text command for descriptions of the options.

1.40 MUIRexx.guide/cycle

-- Command: cycle ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, ATTRS/K/M, LABEL=LABELS/K/F

Cycle gadget objects are created with this command. Cycle and radio gadgets are essentially identical with the only difference being the base object class to create each type of object. All options are identical for these objects.

* ID [I..] - an id can be assigned to a gadget for later reference. The id can be any combination of up to 5 characters. If the id is given without any other arguments, and the gadget has been previously created, then the currently selected label will be returned in RESULT (if OPTIONS RESULTS is specified in the script).

* COMMAND [I..] - if given, the command will be executed whenever the gadget is selected. For a complete description of this option see Command-Port Options. Note that the gadget active label (see the LABELS argument description) will be used to replace a '%s' contained in the command string.

* PORT [I..] - a specific host port may be specified by this argument. For a complete description of this option see Command-Port Options.

* HELP [I..] - with this argument help text may be defined which will be displayed as balloon help whenever the pointer is over the associated gadget. Of course, this is dependant on whether the user set up balloon help in the MUI preference settings.

* NODE [I..] - this argument is used to specify a node in the guide file given in the command line argument HELP for 'MUIRexx'. If the user positions the mouse pointer over the gadget and presses the help button on the keyboard then the guide file will be displayed at the node location.

* LABELS [ISG] - a series of strings (separated by commas) may be specified by this argument. These strings are used as the labels for the gadget object. The currently displayed label may be retrieved by issuing the gadget command with an existing ID. Also, the selected label may be set by issuing the gadget command with an existing ID and label. Note that the labels may contain any of the special formatting

sequences (see MUI Format Sequences).

* ATTRS [ISG] - with this option any MUIA attribute TAGs may be set or retrieved (see Attribute TAGs).

Some useful TAGs for use with this command are:

```
TAG_Name =                          TAG_id          Flags Type

Cycle_Active =                      0x80421788 /* V4  isg LONG */
Radio_Active =                      0x80429b41 /* V4  isg LONG */
ControlChar =                       0x8042120b /* V4  isg char */
CycleChain =                        0x80421ce7 /* V11 isg LONG */
Disabled =                          0x80423661 /* V4  isg BOOL */
FixHeight =                         0x8042a92b /* V4  i.. LONG */
FixHeightTxt =                      0x804276f2 /* V4  i.. STRPTR */
FixWidth =                          0x8042a3f1 /* V4  i.. LONG */
FixWidthTxt =                       0x8042d044 /* V4  i.. STRPTR */
HorizDisappear =                    0x80429615 /* V11 isg LONG */
HorizWeight =                       0x80426db9 /* V4  isg WORD */
MaxHeight =                         0x804293e4 /* V11 i.. LONG */
MaxWidth =                          0x8042f112 /* V11 i.. LONG */
Selected =                          0x8042654b /* V4  isg BOOL */
ShowMe =                            0x80429ba8 /* V4  isg BOOL */
VertDisappear =                     0x8042d12f /* V11 isg LONG */
VertWeight =                        0x804298d0 /* V4  isg WORD */
Weight =                            0x80421d1f /* V4  i.. WORD */
```

Note: Consult MUI:Developer/Autodocs/MUI_Cycle.doc, MUI_Radio.doc, MUI_Area.doc and MUI:Developer/C/Include/libraries/mui.h

Example use of this command:

```
window ID PAGE TITLE '"Character Definition"'
    group HORIZ
        group
            label SINGLE 'Name:'
            label SINGLE 'Sex:'
        endgroup
        group
            string ID NAME CONTENT 'Frodo'
            cycle ID SEX LABELS 'male,female'
        endgroup
    endgroup
    space 2
    group REGISTER LABELS 'Race,Class,Armor,Level'
        group FRAME
            radio ID RACE LABELS 'Human,Elf,Dwarf,Hobbit,Gnome'
        endgroup
        .
        .
        .
endwindow

cycle ID SEX
say result
radio ID RACE
```

```
say result
```

1.41 MUIRexx.guide/radio

```
-- Command: radio ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, ATTRS/K/M,
        LABEL=LABELS/K/F
```

Radio gadget objects are created with this command. Cycle and radio gadgets are essentially identical with the only difference being the base object class to create each type of object. All options are identical for these objects. Refer to the cycle command for descriptions of the options.

1.42 MUIRexx.guide/string

```
-- Command: string ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, ATTRS/K/M,
        CONTENT/K/F
```

String gadget objects are created with this command. String and popasl gadgets are essentially identical with the only difference being the base object class to create each type of object. All options are identical for these objects.

* ID [I..] - an id can be assigned to a string gadget for later reference. The id can be any combination of up to 5 characters. If the id is given without any other arguments, and the string gadget has been previously created, then the current string gadget content will be returned in RESULT (if OPTIONS RESULTS is specified in the script).

* COMMAND [I..] - if given, the command will be executed whenever a string is entered (i.e. a carriage return is hit while the gadget is active). For a complete description of this option see Command-Port Options. Note that the string gadget content (see the CONTENT argument description) will be used to replace a '%s' contained in the command string.

* PORT [I..] - a specific host port may be specified by this argument. For a complete description of this option see Command-Port Options.

* HELP [I..] - with this argument help text may be defined which will be displayed as balloon help whenever the pointer is over the associated string gadget. Of course, this is dependant on whether the user set up balloon help in the MUI preference settings.

* NODE [I..] - this argument is used to specify a node in the guide file given in the command line argument HELP for 'MUIRexx'. If the user positions the mouse pointer over the

string gadget and presses the help button on the keyboard
then the guide file will be displayed at the node location.

* CONTENT [ISG] - the contents of the string gadget is given by this argument.

* ATTRS [ISG] - with this option any MUIA attribute TAGs may be set or retrieved (see Attribute TAGs).

Some useful TAGs for use with this command are:

```
TAG_Name =                          TAG_id          Flags Type

String_Accept =                     0x8042e3e1 /* V4  isg STRPTR */
String_AdvanceOnCR =                0x804226de /* V11 isg BOOL */
String_BufferPos =                  0x80428b6c /* V4  .sg LONG */
String_Contents =                   0x80428ffd /* V4  isg STRPTR */
String_DisplayPos =                 0x8042ccbf /* V4  .sg LONG */
String_Format =                     0x80427484 /* V4  i.g LONG */
String_Integer =                    0x80426e8a /* V4  isg ULONG */
String_MaxLen =                     0x80424984 /* V4  i.g LONG */
String_Reject =                     0x8042179c /* V4  isg STRPTR */
String_Secret =                     0x80428769 /* V4  i.g BOOL */
ControlChar =                       0x8042120b /* V4  isg char */
CycleChain =                        0x80421ce7 /* V11 isg LONG */
Disabled =                          0x80423661 /* V4  isg BOOL */
FixHeight =                         0x8042a92b /* V4  i.. LONG */
FixHeightTxt =                      0x804276f2 /* V4  i.. STRPTR */
FixWidth =                          0x8042a3f1 /* V4  i.. LONG */
FixWidthTxt =                       0x8042d044 /* V4  i.. STRPTR */
HorizDisappear =                    0x80429615 /* V11 isg LONG */
HorizWeight =                       0x80426db9 /* V4  isg WORD */
MaxHeight =                         0x804293e4 /* V11 i.. LONG */
MaxWidth =                          0x8042f112 /* V11 i.. LONG */
Selected =                          0x8042654b /* V4  isg BOOL */
ShowMe =                            0x80429ba8 /* V4  isg BOOL */
VertDisappear =                     0x8042d12f /* V11 isg LONG */
VertWeight =                        0x804298d0 /* V4  isg WORD */
Weight =                            0x80421d1f /* V4  i.. WORD */
```

Note: Consult MUI:Developer/Autodocs/MUI_String.doc,
MUI_Area.doc and MUI:Developer/C/Include/libraries/mui.h

Example use of this command:

```
window ID MAIN TITLE '"ShowIcon"'
    group HORIZ
        button '"parent"' WEIGHT 0 COMMAND '"showicon /"'
        string ID STRG COMMAND '"dirlist ID LIST PATH %s"' PORT SHOW
    endgroup
    .
    .
    .
    popasl ID 104 HELP '"this is an example popasl gadget"'
endwindow

string ID STRG
```

```
say result
```

1.43 MUIRexx.guide/popasl

```
-- Command: popasl ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, ATTRS/K/M,
        CONTENT/K/F, SPEC/K
```

Popasl gadget objects are created with this command. String and popasl gadgets are essentially identical with the only difference being the base object class to create each type of object. The most notable difference is that the popasl object has an attached button that allows the user to bring up an ASL file requestor. The selected file is placed into the string content. All options are identical for these objects with the exception of the SPEC option (used to specify the popbutton image specification). Refer to the string command for descriptions of these options.

* SPEC [I..] - this argument is used to specify a MUI image specification (see MUI Image Specifications) for the popbutton. If not given then the image 'MUII_PopUp' will be used.

1.44 MUIRexx.guide/poplist

```
-- Command: poplist ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K,
        ATTRS/K/M, CONTENT/K/F, SPEC/K, LABELS/K
```

Poplist gadget objects are created with this command. String and poplist gadgets are essentially identical with the only difference being the base object class to create each type of object. The most notable difference is that the poplist object has an attached button that allows the user to bring up a list of strings (specified with the LABELS option). All options are identical for these objects with the exception of the SPEC option (used to specify the popbutton image specification) and the LABELS option. Refer to the string command for descriptions of these options.

* SPEC [I..] - this argument is used to specify a MUI image specification (see MUI Image Specifications) for the popbutton. If not given then the image 'MUII_PopUp' will be used.

* LABELS [ISG] - a series of strings (separated by commas) may be specified by this argument. These strings are used as the labels for the poplist object.

1.45 MUIRexx.guide/slider

```
-- Command: slider ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, ATTRS/K/M
    Slider gadget objects are created with this command.  Slider,
    popslider and knob gadgets are essentially identical with the only
    difference being the base object class to create each type of
    object.  All options are identical for these objects.

       * ID [I..] - an id can be assigned to a slider gadget for later
         reference.  The id can be any combination of up to 5
         characters.  If the id is given without any other arguments,
         and the slider gadget has been previously created, then the
         current slider level will be returned in RESULT (if OPTIONS
         RESULTS is specified in the script).

       * COMMAND [I..] - if given, the command will be executed
         whenever the slider gadget is changed.  For a complete
         description of this option see Command-Port Options.  Note
         that the slider gadget level will be used to replace a '%s'
         contained in the command string.  Caution must be taken since
         movement of the slider can result in a lot of commands.  If
         the command is at all time consuming the result will be very
         sluggish slider action.

       * PORT [I..] - a specific host port may be specified by this
         argument. For a complete description of this option see
         Command-Port Options.

       * HELP [I..] - with this argument help text may be defined
         which will be displayed as balloon help whenever the pointer
         is over the associated slider gadget.  Of course, this is
         dependant on whether the user set up balloon help in the MUI
         preference settings.

       * NODE [I..] - this argument is used to specify a node in the
         guide file given in the command line argument HELP for
         'MUIRexx'.  If the user positions the mouse pointer over the
         slider gadget and presses the help button on the keyboard
         then the guide file will be displayed at the node location.

       * ATTRS [ISG] - with this option any MUIA attribute TAGs may be
         set or retrieved (see Attribute TAGs).

    Some useful TAGs for use with this command are:

      TAG_Name =                        TAG_id          Flags Type

      Slider_Horiz =                    0x8042fad1 /* V11 isg BOOL */
      Slider_Level =                    0x8042ae3a /* V4  isg LONG */
      Slider_Max =                      0x8042d78a /* V4  isg LONG */
      Slider_Min =                      0x8042e404 /* V4  isg LONG */
      Slider_Quiet =                    0x80420b26 /* V6  i.. BOOL */
      Slider_Reverse =                  0x8042f2a0 /* V4  isg BOOL */
      Numeric_Default =                 0x804263e8 /* V11 isg LONG */
      Numeric_Format =                  0x804263e9 /* V11 isg STRPTR */
      Numeric_Max =                     0x8042d78a /* V11 isg LONG */
```

```
        Numeric_Min =                   0x8042e404 /* V11 isg LONG */
        Numeric_Reverse =               0x8042f2a0 /* V11 isg BOOL */
        Numeric_RevLeftRight =          0x804294a7 /* V11 isg BOOL */
        Numeric_RevUpDown =             0x804252dd /* V11 isg BOOL */
        Numeric_Value =                 0x8042ae3a /* V11 isg LONG */
        ControlChar =                   0x8042120b /* V4  isg char */
        CycleChain =                    0x80421ce7 /* V11 isg LONG */
        Disabled =                      0x80423661 /* V4  isg BOOL */
        FixHeight =                     0x8042a92b /* V4  i.. LONG */
        FixHeightTxt =                  0x804276f2 /* V4  i.. STRPTR */
        FixWidth =                      0x8042a3f1 /* V4  i.. LONG */
        FixWidthTxt =                   0x8042d044 /* V4  i.. STRPTR */
        HorizDisappear =                0x80429615 /* V11 isg LONG */
        HorizWeight =                   0x80426db9 /* V4  isg WORD */
        MaxHeight =                     0x804293e4 /* V11 i.. LONG */
        MaxWidth =                      0x8042f112 /* V11 i.. LONG */
        Selected =                      0x8042654b /* V4  isg BOOL */
        ShowMe =                        0x80429ba8 /* V4  isg BOOL */
        VertDisappear =                 0x8042d12f /* V11 isg LONG */
        VertWeight =                    0x804298d0 /* V4  isg WORD */
        Weight =                        0x80421d1f /* V4  i.. WORD */

        Note: Consult MUI:Developer/Autodocs/MUI_Slider.doc, MUI_Numeric.doc,
              MUI_Area.doc and MUI:Developer/C/Include/libraries/mui.h

    Example use of this command:

        window ID DEMO
            .
            .
            .
            group HORIZ
                space HORIZ
                group
                    knob ID KNOB HELP '"an example knob gadget"'
                    popslider ID PSLD HELP '"an example popup slider gadget"'
                endgroup
                meter ID METR NODE '"meter"' LABEL '"meter"'
                space HORIZ
            endgroup
            slider ID SLDR ATTRS Slider_Level 50
            gauge ID GAUG NODE '"gauge"' LABEL '"level %ld"' ATTRS Gauge_Horiz  ↩
               TRUE
            object CLASS '"Scale.mui"'
        endwindow
```

1.46 MUIRexx.guide/popslider

```
 -- Command: popslider ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K,
        ATTRS/K/M
    Popslider gadget objects are created with this command.  Slider,
    popslider and knob gadgets are essentially identical with the only
    difference being the base object class to create each type of
```

object. Specifically, while unselected the gadget displays the current numeric value in a button. If selected then a slider pops up allowing selection of a new value. All options are identical for these objects. Refer to the slider command for descriptions of the options.

1.47 MUIRexx.guide/knob

-- Command: knob ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, ATTRS/K/M
Knob gadget objects are created with this command. Slider, popslider and knob gadgets are essentially identical with the only difference being the base object class to create each type of object. All options are identical for these objects. Refer to the slider command for descriptions of the options.

1.48 MUIRexx.guide/list

-- Command: list ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, TITLE/K, POS/K, INSERT/S, REMOVE/S, NODUP/S, TOGGLE/S, ATTRS/K/M, STRING=STRINGS/K/F
List objects are created with this command.

* ID [I..] - an id can be assigned to a list for later reference. The id can be any combination of up to 5 characters. If the id is given without any other arguments, and the list object has been previously created, then the currently selected line will be returned in RESULT (if OPTIONS RESULTS is specified in the script). If multiple lines are selected then each line will be returned with each list command. The line entry in the list will be deselected. A null string ("") will be returned if no lines are selected (or the last selected line has been reached).

* COMMAND [I..] - if given, the command will be executed whenever a line in the list is double clicked. For a complete description of this option see Command-Port Options. Note that the selected line will be used to replace a ‘%s’ contained in the command string.

* PORT [I..] - a specific host port may be specified by this argument. For a complete description of this option see Command-Port Options.

* HELP [I..] - with this argument help text may be defined which will be displayed as balloon help whenever the pointer is over the associated list. Of course, this is dependant on whether the user set up balloon help in the MUI preference settings.

* NODE [I..] - this argument is used to specify a node in the guide file given in the command line argument HELP for 'MUIRexx'. If the user positions the mouse pointer over the list and presses the help button on the keyboard then the guide file will be displayed at the node location.

* TITLE [IS.] - this argument is used to specify a title for the list. A title will be shown at the top of the list and will remain even if the list scrolled. Note that the format of the title will be the same as for the string entries (see the STRING option description below).

* POS [.SG] - if this argument is given and a string is specified (with the STRING argument) then the string will be inserted at this position. Special values (see List.mui Autodoc entry for MUIA_List_Insert) may be used. If no string is given then the string located at the given position will be returned in the 'ARexx' variable RESULT. If a null string is given (by specifying the STRING option with no string or "") then the string at this position will be removed.

* INSERT [.S.] - if this switch is given then any string supplied by the STRING argument will be inserted into the current list. If no position is specified (with the POS argument) then the string will be inserted at the current DropMark (see List.mui Autodoc entry for MUIA_List_DropMark).

* REMOVE [.S.] - if this switch is given then an entry will be removed from the list. If no position or label is specified (with the POS argument) then the currently selected entry will be removed. If a label is given (using the LABEL option) then the entry matching that label will be removed.

* NODUP [IS.] - if this switch is given then no duplicate strings will be displayed.

* TOGGLE [.S.] - if this switch is given then the select state of each displayed string will be toggled.

* STRING [ISG] - a string to be entered into the list may be specified by this argument. Note that the string may contain any of the special formatting sequences (see MUI Format Sequences). Additionally, multicolumn lists may be created. A string may, in fact, consist of several strings separated with commas. If the list is given a format (with the LIST_FORMAT attribute TAG) then these strings will be displayed in the appropriate column (as defined by the format, see MUI List Format). Prepending any substring with an equals symbol (=) will force the rest of the string to be interpreted literally thereby allowing commas within a string. Note that the equals symbol will be removed from any retreived literal string.

* ATTRS [ISG] - with this option any MUIA attribute TAGs may be set or retrieved (see Attribute TAGs).

Some useful TAGs for use with this command are:

```
     TAG_Name =                          TAG_id          Flags Type

     List_Active =                       0x8042391c /* V4  isg LONG */
     List_AdjustHeight =                 0x8042850d /* V4  i.. BOOL */
     List_AdjustWidth =                  0x8042354a /* V4  i.. BOOL */
     List_AutoVisible =                  0x8042a445 /* V11 isg BOOL */
     List_DragSortable =                 0x80426099 /* V11 isg BOOL */
     List_DropMark =                     0x8042aba6 /* V11 ..g LONG */
     List_Entries =                      0x80421654 /* V4  ..g LONG */
     List_First =                        0x804238d4 /* V4  ..g LONG */
     List_Format =                       0x80423c0a /* V4  isg STRPTR */
     List_InsertPosition =               0x8042d0cd /* V9  ..g LONG */
     List_MinLineHeight =                0x8042d1c3 /* V4  i.. LONG */
     List_Quiet =                        0x8042d8c7 /* V4  .s. BOOL */
     List_ShowDropMarks =                0x8042c6f3 /* V11 isg BOOL */
     List_Title =                        0x80423e66 /* V6  isg char * */
     List_Visible =                      0x8042191f /* V4  ..g LONG */
     Listview_ClickColumn =              0x8042d1b3 /* V7  ..g LONG */
     Listview_DefClickColumn =           0x8042b296 /* V7  isg LONG */
     Listview_DoubleClick =              0x80424635 /* V4  i.g BOOL */
     Listview_DragType =                 0x80425cd3 /* V11 isg LONG */
     Listview_Input =                    0x8042682d /* V4  i.. BOOL */
     Listview_MultiSelect =              0x80427e08 /* V7  i.. LONG */
     Listview_ScrollerPos =              0x8042b1b4 /* V10 i.. BOOL */
     Listview_SelectChange =             0x8042178f /* V4  ..g BOOL */
     CycleChain =                        0x80421ce7 /* V11 isg LONG */
     Disabled =                          0x80423661 /* V4  isg BOOL */
     HorizDisappear =                    0x80429615 /* V11 isg LONG */
     HorizWeight =                       0x80426db9 /* V4  isg WORD */
     ShowMe =                            0x80429ba8 /* V4  isg BOOL */
     VertDisappear =                     0x8042d12f /* V11 isg LONG */
     VertWeight =                        0x804298d0 /* V4  isg WORD */
     Weight =                            0x80421d1f /* V4  i.. WORD */

     Note: Consult MUI:Developer/Autodocs/MUI_List.doc, MUI_Listview.doc,
           MUI_Area.doc and MUI:Developer/C/Include/libraries/mui.h
```

Example use of this command:

```
     window ID DEMO
         list ID ALST ATTRS List_Format "MIW=25 BAR,MIW=25 BAR,MIW=25"'
         list ID BLST
         .
         .
         .
     endwindow
     list ID ALST INSERT POS 0 STRING '"column 1,column 2,column 3"'
     list ID BLST INSERT POS 0 STRING '"=this one, that one"'
```

1.49 MUIRexx.guide/dirlist

```
-- Command: dirlist ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K, PATH/K,
         PATTERN/K, REREAD/S ,TOGGLE/S, ATTRS/K/M
    Dirlist objects are created with this command.

       * ID [I..] - an id can be assigned to a dirlist for later
         reference. The id can be any combination of up to 5
         characters.  If the id is given without any other arguments,
         and the dirlist has been previously created, then the
         currently selected file (with path) will be returned in
         RESULT (if OPTIONS RESULTS is specified in the script).  If
         multiple files are selected then each file name (with path)
         will be returned with each dirlist command.  The file name
         entry in the list will be deselected.  A null string ("")
         will be returned if no files are selected (or the last
         selected file has been reached).

       * COMMAND [I..] - if given, the command will be executed
         whenever an item in the dirlist is double clicked.  For a
         complete description of this option see Command-Port Options.
          Note that the full path of the selected item will be used to
         replace a '%s' contained in the command string.

       * PORT [I..] - a specific host port may be specified by this
         argument. For a complete description of this option see
         Command-Port Options.

       * HELP [I..] - with this argument help text may be defined
         which will be displayed as balloon help whenever the pointer
         is over the associated dirlist. Of course, this is dependant
         on whether the user set up balloon help in the MUI preference
         settings.

       * NODE [I..] - this argument is used to specify a node in the
         guide file given in the command line argument HELP for
         'MUIRexx'. If the user positions the mouse pointer over the
         dirlist and presses the help button on the keyboard then the
         guide file will be displayed at the node location.

       * PATH [ISG] - at creation this argument specifies the initial
         directory path. When the dirlist command is issued with just
         the ID argument a fully qualified path name is returned for
         the file or directory selected in the listview.

       * PATTERN [IS.] - this argument sets the accept pattern for the
         directory list.  Any standard AmigaDOS pattern may be given.
         Note that if a path is set (see PATH argument) or the
         directory is reread (see REREAD argument) then this pattern
         will be reflected.

       * REREAD [.S.] - if this switch is given then the dirlist will
         be updated with the current directory.

       * TOGGLE [.S.] - if this switch is given then the select state
         of each displayed file will be toggled.
```

* ATTRS [ISG] - with this option any MUIA attribute TAGs may be set or retrieved (see Attribute TAGs).

Some useful TAGs for use with this command are:

```
TAG_Name =                          TAG_id          Flags Type

Dirlist_Directory =                 0x8042ea41 /* V4  isg STRPTR */
Dirlist_DrawersOnly =               0x8042b379 /* V4  is. BOOL */
Dirlist_FilesOnly =                 0x8042896a /* V4  is. BOOL */
Dirlist_FilterDrawers =             0x80424ad2 /* V4  is. BOOL */
Dirlist_MultiSelDirs =              0x80428653 /* V6  is. BOOL */
Dirlist_NumBytes =                  0x80429e26 /* V4  ..g LONG */
Dirlist_NumDrawers =                0x80429cb8 /* V4  ..g LONG */
Dirlist_NumFiles =                  0x8042a6f0 /* V4  ..g LONG */
Dirlist_RejectIcons =               0x80424808 /* V4  is. BOOL */
Dirlist_SortDirs =                  0x8042bbb9 /* V4  is. LONG */
Dirlist_SortHighLow =               0x80421896 /* V4  is. BOOL */
Dirlist_SortType =                  0x804228bc /* V4  is. LONG */
Dirlist_Status =                    0x804240de /* V4  ..g LONG */
List_Active =                       0x8042391c /* V4  isg LONG */
List_AdjustHeight =                 0x8042850d /* V4  i.. BOOL */
List_AdjustWidth =                  0x8042354a /* V4  i.. BOOL */
List_AutoVisible =                  0x8042a445 /* V11 isg BOOL */
List_DragSortable =                 0x80426099 /* V11 isg BOOL */
List_DropMark =                     0x8042aba6 /* V11 ..g LONG */
List_Entries =                      0x80421654 /* V4  ..g LONG */
List_First =                        0x804238d4 /* V4  ..g LONG */
List_Format =                       0x80423c0a /* V4  isg STRPTR */
List_InsertPosition =               0x8042d0cd /* V9  ..g LONG */
List_MinLineHeight =                0x8042d1c3 /* V4  i.. LONG */
List_Quiet =                        0x8042d8c7 /* V4  .s. BOOL */
List_ShowDropMarks =                0x8042c6f3 /* V11 isg BOOL */
List_Title =                        0x80423e66 /* V6  isg char * */
List_Visible =                      0x8042191f /* V4  ..g LONG */
Listview_ClickColumn =              0x8042d1b3 /* V7  ..g LONG */
Listview_DefClickColumn =           0x8042b296 /* V7  isg LONG */
Listview_DoubleClick =              0x80424635 /* V4  i.g BOOL */
Listview_DragType =                 0x80425cd3 /* V11 isg LONG */
Listview_Input =                    0x8042682d /* V4  i.. BOOL */
Listview_MultiSelect =              0x80427e08 /* V7  i.. LONG */
Listview_ScrollerPos =              0x8042b1b4 /* V10 i.. BOOL */
Listview_SelectChange =             0x8042178f /* V4  ..g BOOL */
CycleChain =                        0x80421ce7 /* V11 isg LONG */
Disabled =                          0x80423661 /* V4  isg BOOL */
HorizDisappear =                    0x80429615 /* V11 isg LONG */
HorizWeight =                       0x80426db9 /* V4  isg WORD */
ShowMe =                            0x80429ba8 /* V4  isg BOOL */
VertDisappear =                     0x8042d12f /* V11 isg LONG */
VertWeight =                        0x804298d0 /* V4  isg WORD */
Weight =                            0x80421d1f /* V4  i.. WORD */
```

Note: Consult MUI:Developer/Autodocs/MUI_List.doc, MUI_Listview.doc, MUI_Dirlist.doc, MUI_Area.doc and MUI:Developer/C/Include/libraries/mui.h

```
Example use of this command:

     window TITLE '"MUIRexx Demo"' COMMAND '"quit"' PORT DEMO
         dirlist ID DIR1 PATH '"ram:"' PRESS APP DROP,
             COMMAND '"dirlist ID DIR1 PATH %s"' PORT DEMO NODE '"dirlist"',
             ATTRS Frame Frame_Text Listview_DragType  ↩
                 Listview_DragType_Immediate
         .
         .
         .
     endwindow

     dirlist ID DIR1 ATTRS Dirlist_Directory
     say result
```

1.50 MUIRexx.guide/volumelist

```
-- Command: volumelist ID/K, COMMAND/K, PORT/K, HELP/K, NODE/K,
        ATTRS/K/M
    volumelist objects are created with this command.

       * ID [I..] - an id can be assigned to a volumelist for later
         reference. The id can be any combination of up to 5
         characters.  If the id is given without any other arguments,
         and the volumelist has been previously created, then the
         currently selected volume will be returned in RESULT (if
         OPTIONS RESULTS is specified in the script).

       * COMMAND [I..] - if given, the command will be executed
         whenever an item in the volumelist is double clicked.  For a
         complete description of this option see Command-Port Options.
          Note that the selected volume name will be used to replace a
         '%s' contained in the command string.

       * PORT [I..] - a specific host port may be specified by this
         argument. For a complete description of this option see
         Command-Port Options.

       * HELP [I..] - with this argument help text may be defined
         which will be displayed as balloon help whenever the pointer
         is over the associated volumelist. Of course, this is
         dependant on whether the user set up balloon help in the MUI
         preference settings.

       * NODE [I..] - this argument is used to specify a node in the
         guide file given in the command line argument HELP for
         'MUIRexx'.  If the user positions the mouse pointer over the
         volumelist and presses the help button on the keyboard then
         the guide file will be displayed at the node location.

       * ATTRS [ISG] - with this option any MUIA attribute TAGs may be
         set or retrieved (see Attribute TAGs).
```

Some useful TAGs for use with this command are:

```
TAG_Name =                          TAG_id          Flags Type

List_Active =                       0x8042391c /* V4  isg LONG */
List_AdjustHeight =                 0x8042850d /* V4  i.. BOOL */
List_AdjustWidth =                  0x8042354a /* V4  i.. BOOL */
List_AutoVisible =                  0x8042a445 /* V11 isg BOOL */
List_DragSortable =                 0x80426099 /* V11 isg BOOL */
List_DropMark =                     0x8042aba6 /* V11 ..g LONG */
List_Entries =                      0x80421654 /* V4  ..g LONG */
List_First =                        0x804238d4 /* V4  ..g LONG */
List_Format =                       0x80423c0a /* V4  isg STRPTR */
List_InsertPosition =               0x8042d0cd /* V9  ..g LONG */
List_MinLineHeight =                0x8042d1c3 /* V4  i.. LONG */
List_Quiet =                        0x8042d8c7 /* V4  .s. BOOL */
List_ShowDropMarks =                0x8042c6f3 /* V11 isg BOOL */
List_Title =                        0x80423e66 /* V6  isg char * */
List_Visible =                      0x8042191f /* V4  ..g LONG */
Listview_ClickColumn =              0x8042d1b3 /* V7  ..g LONG */
Listview_DefClickColumn =           0x8042b296 /* V7  isg LONG */
Listview_DoubleClick =              0x80424635 /* V4  i.g BOOL */
Listview_DragType =                 0x80425cd3 /* V11 isg LONG */
Listview_Input =                    0x8042682d /* V4  i.. BOOL */
Listview_MultiSelect =              0x80427e08 /* V7  i.. LONG */
Listview_ScrollerPos =              0x8042b1b4 /* V10 i.. BOOL */
Listview_SelectChange =             0x8042178f /* V4  ..g BOOL */
CycleChain =                        0x80421ce7 /* V11 isg LONG */
Disabled =                          0x80423661 /* V4  isg BOOL */
HorizDisappear =                    0x80429615 /* V11 isg LONG */
HorizWeight =                       0x80426db9 /* V4  isg WORD */
ShowMe =                            0x80429ba8 /* V4  isg BOOL */
VertDisappear =                     0x8042d12f /* V11 isg LONG */
VertWeight =                        0x804298d0 /* V4  isg WORD */
Weight =                            0x80421d1f /* V4  i.. WORD */
```

Note: Consult MUI:Developer/Autodocs/MUI_List.doc, MUI_Listview.doc, MUI_Volumelist.doc, MUI_Area.doc and MUI:Developer/C/Include/libraries/mui.h

Example use of this command:

```
window TITLE '"MUIRexx Demo"' COMMAND '"quit"' PORT DEMO
    volumelist,
        COMMAND '"dirlist ID DIR1 PATH %s"' PORT DEMO NODE '"volumelist ↩
            "',
        ATTRS Weight 50
    .
    .
    .
endwindow
```

1.51 MUIRexx.guide/object

```
-- Command: object ID/K, HELP/K, NODE/K, CLASS/K, BOOPSI/S, ATTRS/K/M
```

Objects from MUI internal, external and BOOPSI classes are created with this command. Note that while this is a very powerful and flexible command it can easily result in the unexpected, including system crashes, so beware.

* ID [I..] - an id can be assigned to a object for later reference. The id can be any combination of up to 5 characters.

* HELP [I..] - with this argument help text may be defined which will be displayed as balloon help whenever the pointer is over the associated object. Of course, this is dependant on whether the user set up balloon help in the MUI preference settings.

* NODE [I..] - this argument is used to specify a node in the guide file given in the command line argument HELP for 'MUIRexx'. If the user positions the mouse pointer over the object and presses the help button on the keyboard then the guide file will be displayed at the node location.

* CLASS [I..] - this argument allows specification of the object class. The class may be any internal or external MUI or BOOPSI (see the BOOPSI option) gadget class.

* BOOPSI [I..] - if this switch is given then the object will be created from a BOOPSI gadget class. BOOPSI objects will, by default, have the following TAG ids and values set:

TAG	value
GA_Left	0
GA_Top	0
GA_Width	0
GA_Height	0
ICA_TARGET	ICTARGET_IDCMP

* ATTRS [ISG] - with this option any MUIA attribute TAGs may be set or retrieved (see Attribute TAGs).

Some useful TAGs for use with this command are:

```
TAG_Name =                          TAG_id          Flags Type

Boopsi_MaxHeight =                  0x8042757f /* V4  isg ULONG */
Boopsi_MaxWidth =                   0x8042bcb1 /* V4  isg ULONG */
Boopsi_MinHeight =                  0x80422c93 /* V4  isg ULONG */
Boopsi_MinWidth =                   0x80428fb2 /* V4  isg ULONG */
Boopsi_Remember =                   0x8042f4bd /* V4  i.. ULONG */
Boopsi_Smart =                      0x8042b8d7 /* V9  i.. BOOL */
Boopsi_TagDrawInfo =                0x8042bae7 /* V4  isg ULONG */
Boopsi_TagScreen =                  0x8042bc71 /* V4  isg ULONG */
Boopsi_TagWindow =                  0x8042e11d /* V4  isg ULONG */
```

```
ControlChar =                    0x8042120b /* V4  isg char */
CycleChain =                     0x80421ce7 /* V11 isg LONG */
Disabled =                       0x80423661 /* V4  isg BOOL */
Draggable =                      0x80420b6e /* V11 isg BOOL */
FillArea =                       0x804294a3 /* V4  is. BOOL */
FixHeight =                      0x8042a92b /* V4  i.. LONG */
FixHeightTxt =                   0x804276f2 /* V4  i.. STRPTR */
FixWidth =                       0x8042a3f1 /* V4  i.. LONG */
FixWidthTxt =                    0x8042d044 /* V4  i.. STRPTR */
HorizDisappear =                 0x80429615 /* V11 isg LONG */
HorizWeight =                    0x80426db9 /* V4  isg WORD */
MaxHeight =                      0x804293e4 /* V11 i.. LONG */
MaxWidth =                       0x8042f112 /* V11 i.. LONG */
Selected =                       0x8042654b /* V4  isg BOOL */
ShowMe =                         0x80429ba8 /* V4  isg BOOL */
ShowSelState =                   0x8042caac /* V4  i.. BOOL */
VertDisappear =                  0x8042d12f /* V11 isg LONG */
VertWeight =                     0x804298d0 /* V4  isg WORD */
Weight =                         0x80421d1f /* V4  i.. WORD */
```

Note: Consult MUI:Developer/Autodocs/MUI_Boopsi.doc, MUI_Area.doc and MUI:Developer/C/Include/libraries/mui.h

Example use of this command:

```
WHEEL_Hue =                      0x84000001
WHEEL_Saturation =               0x84000002
WHEEL_Screen =                   0x84000009

window TITLE '"MUIRexx Demo"' COMMAND '"quit"' PORT DEMO
    group HORIZ
        group
            label DOUBLE '"Hue:"'
            label DOUBLE '"Saturation:"'
        endgroup
        group
            gauge ID HUE ATTRS Gauge_Max 16384,
                               Gauge_Divide 262144,
                               Gauge_Horiz TRUE
            gauge ID SAT ATTRS Gauge_Max 16384,
                               Gauge_Divide 262144,
                               Gauge_Horiz TRUE
        endgroup
    endgroup
    object ID BOOP BOOPSI CLASS '"colorwheel.gadget"',
        ATTRS Boopsi_MinWidth 30,
              Boopsi_MinHeight 30,
              Boopsi_Remember WHEEL_Hue,
              Boopsi_Remember WHEEL_Saturation,
              Boopsi_TagScreen WHEEL_Screen,
              WHEEL_Screen 0,
              WHEEL_Saturation 0,
              FillArea TRUE
    .
    .
    .
endwindow
```

```
          method ID BOOP Notify WHEEL_Hue EveryTime,
                         @HUE 4 Set Gauge_Current TriggerValue
          method ID BOOP Notify WHEEL_Saturation EveryTime,
                         @SAT 4 Set Gauge_Current TriggerValue
```

1.52 MUIRexx.guide/Misc

```
Misc
====

   These commands don't fit any of the previous categories so here they
are.


  request
  aslrequest
  callhook
  method
  setvar
  getvar
  application
  monitor
  print
```

1.53 MUIRexx.guide/request

```
 -- Command: request ID/K, TITLE/K, GADGETS/K, FILE/K, STRING/F
     This command will bring up a standard 'MUI' requester.  Note that
     this command is syncronous.  That is once it is issued it will not
     return a result until the user has selected a gadget.  Once a
     gadget has been selected then a number will be returned in the
     'ARexx' variable RESULT (assuming 'options results' was specified
     in the script). The first gadget will return a 1, the second a 2,
     and so on.  The last gadget, however, will return a 0 (this is by
     convention since the last gadget is typically a CANCEL or
     equivilent gadget).

        * ID - this argument specifies a window ID.  If specified then
          the requester will be centered in the window.

        * TITLE - this argument specifies the requester title (placed
          in the title bar of the requester window).

        * GADGETS - this argument specifies the gadget labels.  The
          labels are given as a single string with a vertical bar, |
          used to separate each label.  Additionally, each label can
          contain an underscore, _ prior to any character.  The
          character will be the keyboard shortcut to activate the
```

associated gadget.

* FILE - if given, this argument specifies the file to get the contents for the requester. All line breaks in the file will be included. Note that this argument overrides the STRING argument.

* STRING - this argument specifies the string to display in the requester.

Example use of this command:

```
request ID MDIR GADGETS '"OK|Cancel"' '"Delete selected entries?"'
```

1.54 MUIRexx.guide/aslrequest

-- Command: aslrequest ID/K, TITLE/K, ATTRS/K/M

This command will bring up a standard ASL file requester. Note that this command is syncronous. That is once it is issued it will not return a result until the user has selected a file/directory or canceled. Once a file/directory has been selected then the fully path-qualified name will be returned in the 'ARexx' variable RESULT (assuming 'options results' was specified in the script).

* ID - this argument specifies a window ID. If specified then the requester will be associated with the window.

* TITLE - this argument specifies the requester title (placed in the title bar of the requester window).

* ATTRS [ISG] - with this option any ASL attribute TAGs may be set (refer to AmigaDOS RKMs for details).

Some useful TAGs for use with this command are:

TAG_Name =	TAG_id
ASLFR_InitialLeftEdge	0x80080003
ASLFR_InitialTopEdge	0x80080004
ASLFR_InitialWidth	0x80080005
ASLFR_InitialHeight	0x80080006
ASLFR_InitialFile	0x80080008
ASLFR_InitialDrawer	0x80080009
ASLFR_InitialPattern	0x8008000A
ASLFR_DoSaveMode	0x8008002C
ASLFR_DoMultiSelect	0x8008002D
ASLFR_DoPatterns	0x8008002E
ASLFR_DrawersOnly	0x8008002F
ASLFR_RejectIcons	0x8008003C
ASLFR_RejectPattern	0x8008003D
ASLFR_AcceptPattern	0x8008003E
ASLFR_FilterDrawers	0x8008003F

1.55 MUIRexx.guide/callhook

```
-- Command: callhook ID/K, COMMAND/K, PORT/K, PRESS/S, APP/S, DROP/S,
         INCLUDE/K, EXCLUDE/K, TRIG=ATTRS/K/M
    This command is used to define callback notification methods for
    objects. Trigger actions can be defined for object selection, drop
    and app operations, as well as object specific actions.

       * ID - an id of an existing gadget.

       * COMMAND - if given, the command will be executed whenever the
         gadget is pressed (PRESS switch specified or default for no
         switch), an icon is dropped (APP switch), or another object
         is dropped (DROP switch) on the gadget.  For a complete
         description of this option see Command-Port Options.  Note
         that the object label (for PRESS actions), or an icon name
         (for APP actions), or a dropped object label (for DROP
         actions) will be used to replace a '%s' contained in the
         command string, as appropriate.

       * PORT [I..] - a specific host port may be specified by this
         argument. For a complete description of this option see
         Command-Port Options.

       * PRESS - if this flag is given then the specified command
         (given in the COMMAND option) will will issued if the object
         is pressed. (same as the default action given with the
         object's COMMAND option).

       * APP - if this flag is given then the specified command (given
         in the COMMAND option) will will issued if an icon is dropped
         on the object from the Workbench.

       * DROP - if this flag is given then the specified command (given
         in the COMMAND option) will will issued if another object is
         dropped on the object through a drag and drop operation.
         Using the INCLUDE or EXCLUDE options drag and drop operations
         can be restricted to specific objects.

       * INCLUDE - Limits drag and drop operations to a list of
         objects.  The list consists of object ids separated with
         commas.  The specified objects do not have to actually exist
         (i.e. the listed objects can be created after the 'callhook'
         is defined).

       * EXCLUDE - Limits drag and drop operations by excluding a list
         of objects.  The list consists of object ids separated with
         commas.  The specified objects do not have to actually exist
         (i.e.  the listed objects can be created after the 'callhook'
         is defined).

       * TRIG - A TAG pair consisting of a trigger attribute and
```

```
        value.  Used to create arbitrary callback hook triggers.  Do
        not use if PRESS, APP or DROP options are specified.

    Example use of this command:

        /* the following defines a command that will be triggered everytime
           a list entry becomes active */
        callhook ID DIR1 COMMAND """string ID STR1 CONTENT %s""" PORT DIRUTIL,
            TRIG MUIA_List_Active MUIV_EveryTime

        /* the following defines a command triggered by a drop action, but will
           only accept drop querys from the 'BLST' object */
        callhook ID VLST DROP COMMAND """build:attrs VADD %s""" INCLUDE 'BLST'
```

1.56 MUIRexx.guide/method

```
-- Command: method ID/K, ARGS/M
    This command allows construction of class methods (ala the domethod
    function).  While this command is very powerful it is also quite
    complicated and possibly dangerous (can result in system crashes).
     Care should be taken by consulting the autodocs describing the
    method to be constructed.  RESULT will contain the value returned
    by the domethod function.

       * ID - this argument specifies the ID of the reference object
         for the method.  If it is not given then the application
         object will be used.

       * ARGS - these arguments are the remaining arguments passed to
         the domethod function.  The arguments may be TAG ids or TAG
         values (see Attribute TAGs).  TAG values may also be object
         pointers which are specified by preceding an object ID with
         an @.

    Example use of this command:

        method Application_AboutMUI 0       /* bring up about_MUI requester */
        method Application_OpenConfigWindow /* open MUI settings program */

        /* an example notification method */

        method ID BOOP Notify WHEEL_Hue EveryTime,
                       @HUE 4 Set Gauge_Current TriggerValue
```

1.57 MUIRexx.guide/setvar

```
-- Command: setvar NAME/A, VALUE/F
    This command will set an internal 'MUIRexx' variable to any value
```

(stored as a string) which can be retrieved or reset later within the same application (not necessarily the same script). This ability can be used to pass information between 'ARexx' scripts used in the same application. Note that variables may also be used in place of object ID values, in which case the value of the variable will be used as the actual object ID (this value must be a string 5 characters or less, but the variable name can be any length).

* NAME - this argument defines the variable name.

* VALUE - this argument defines the value of the variable and can consist of any characters.

1.58 MUIRexx.guide/getvar

-- Command: getvar NAME/A
This command will retrieve an internal 'MUIRexx' variable. The value will be placed into the 'ARexx' variable RESULT (assuming OPTIONS RESULTS was specified in the calling script).

* NAME - this argument specifies the variable name.

1.59 MUIRexx.guide/application

-- Command: application ATTRS/K/M
This command allows attributes to be set and retrieved for the application object.

* ATTRS [ISG] - with this option any MUIA attribute TAGs may be set or retrieved (see Attribute TAGs).

Some useful TAGs for use with this command are:

```
TAG_Name =                          TAG_id          Flags Type

Application_Active =                0x804260ab /* V4  isg BOOL */
Application_Author =                0x80424842 /* V4  i.g STRPTR */
Application_Base =                  0x8042e07a /* V4  i.g STRPTR */
Application_Copyright =             0x8042ef4d /* V4  i.g STRPTR */
Application_Description =           0x80421fc6 /* V4  i.g STRPTR */
Application_DoubleStart =           0x80423bc6 /* V4  ..g BOOL */
Application_Iconified =             0x8042a07f /* V4  .sg BOOL */
Application_Sleep =                 0x80425711 /* V4  .s. BOOL */
Application_Title =                 0x804281b8 /* V4  i.g STRPTR */
Application_Version =               0x8042b33f /* V4  i.g STRPTR */
```

Note: Consult MUI:Developer/Autodocs/MUI_Application.doc and

```
            MUI:Developer/C/Include/libraries/mui.h
```

Example use of this command:

```
     application ATTRS Application_Iconified TRUE      /* forces  ↩
        iconification */

     application ATTRS Application_Version
     say result
```

1.60 MUIRexx.guide/monitor

-- Command: monitor ON/S, OFF/S, ERROR/S, OUTPUT/F
This command is used to open or close a monitor console window. The monitor, depending on the options given, will display received command lines and errors.

* ON - if given then the console window (or file) will be opened. Both received command lines and error messages will be displayed (or saved). If no options are given then the console open state will be toggled.

* OFF - if given then the console window (or file) will be closed.

* ERROR - if given then the console window (or file) will be opened. Only error messages and the associated command line will be displayed (or saved).

* OUTPUT - this argument is used to specify the console output device or file.

Example use of this command:

```
     monitor on 'cnc:0/660/840/240//auto'
     monitor error 'ram:error.out'
     monitor off
```

1.61 MUIRexx.guide/print

-- Command: print STRING/F
This command is used to output a string to the console previously opened by the monitor command.

* STRING - string to be output.

1.62 MUIRexx.guide/Utilities

```
Utilities
*********


  MUIRexxDir            The little directory utility that could.
  MUIRexxBuild          Using MUIRexx to build MUIRexx apps.
  MUIRexxDock           Dock construction 101.
```

1.63 MUIRexx.guide/MUIRexxDir

```
MUIRexxDir
==========
```

   This utility is an example of a complete application built entirely from ‘ARexx’ scripts and ‘MUIRexx’. Online help is available by positioning the mouse pointer over an object and pressing the [Help] key on the keyboard.

1.64 MUIRexx.guide/MUIRexxBuild

```
MUIRexxBuild
============
```

   This ‘MUIRexx’ application is used to help build other ‘MUIRexx’ applications. Some online help is available along with bubble help to aid in use of the application.

1.65 MUIRexx.guide/MUIRexxDock

```
MUIRexxDock
===========
```

   This ‘MUIRexx’ application is used to build and maintain utility docks. There is currently no documentation available, but hopefully operation is intuitive enough to figure out. Try opening the edit window (using the dock’s menu) and drag and drop either a dock icon or an icon from the Workbench onto the edit window icon area (above the cycle gadget). After editting the dock item either drag the icon back to the dock (replacing the icon it is dropped on) or press the [add] button to add the icon to the end of the dock. Also while the edit

window is open dock items in the dock can be rearranged by dragging one icon onto another. Dock items can be buttons (typically used to start programs), switches (typically used as a toggle), docks (other subdocks of items), or pop groups. Pop groups may consist of any set of objects and are created in the edit window by entering 'MUIRexx' commands into the list located on the 'Pop' register page. Simple buttons may be easily created by dropping command icons from the Workbench onto the 'Pop' list. To edit or delete command lines in the 'Pop' list simply double click on the line and edit the string or press the [del] button to delete. Entering a string will add the command line to the end of the 'Pop' list. Lines in this list are drag sortable. Also, additional commands may be entered into the list located on the 'Add' register page. These additional commands are inserted into the beginning of the group that will contain the dock item. This feature is particularly useful for including context menus with specific dock items.

1.66 MUIRexx.guide/Example Macro

```
Example Macro
*************

     /* A simple example based on one of the demos supplied with MUI */
     options results

     Selected =                        0x8042654b /* V4  isg BOOL */
     Slider_Level =                    0x8042ae3a /* V4  isg LONG */

     TRUE = 1

     address PAGES

     window ID PAGE TITLE '"Character Definition"' COMMAND 'quit' PORT PAGES
         group HORIZ
             group
                 label SINGLE 'Name:'
                 label SINGLE 'Sex:'
             endgroup
             group
                 string ID NAME CONTENT 'Frodo'
                 cycle ID SEX LABELS 'male,female'
             endgroup
         endgroup

         space 2
         group REGISTER LABELS 'Race,Class,Armor,Level'
             group FRAME
                 radio ID RACE LABELS 'Human,Elf,Dwarf,Hobbit,Gnome'
             endgroup
             group FRAME
                 radio ID CLAS LABELS 'Warrior,Rogue,Bard,Monk,Magician,Archmage'
             endgroup
```

```
            group
                group HORIZ
                    group
                        label SINGLE 'Cloak:'
                        label SINGLE 'Shield:'
                        label SINGLE 'Gloves:'
                        label SINGLE 'Helmet:'
                    endgroup
                    group
                        check ID CHK1 ATTRS Selected TRUE
                        check ID CHK2 ATTRS Selected TRUE
                        check ID CHK3 ATTRS Selected TRUE
                        check ID CHK4 ATTRS Selected TRUE
                    endgroup
                endgroup
            endgroup

            group
                group HORIZ
                    group
                        label DOUBLE 'Experience:'
                        label DOUBLE 'Strength:'
                        label DOUBLE 'Dexterity:'
                        label DOUBLE 'Condition:'
                        label DOUBLE 'Intelligence:'
                    endgroup
                    group
                        slider ATTRS Slider_Level 3
                        slider ATTRS Slider_Level 42
                        slider ATTRS Slider_Level 24
                        slider ATTRS Slider_Level 39
                        slider ATTRS Slider_Level 74
                    endgroup
                endgroup
            endgroup

        endgroup
    endwindow
    exit
```

1.67 MUIRexx.guide/Command-Port Options

```
Command/Port Options
********************

   * COMMAND - if given, the command will be executed whenever the
     object is triggered (specific trigger is defined by the object
     type).  The command will be issued to the host port specified by
     the PORT argument.  Note that the command is run asyncronously (as
     a detached process) and only inherits the global path if 'MUIRexx'
     is started from a shell.  The command text may contain a format
```

```
     specifier ('%s') in which case before issuing the command the
     format specifier will be replaced by a label whose content is
     dependant on the object type.

   * PORT - a specific host port may be specified by this argument.  The
     defined command will be issued to this port whenever the object is
     triggered.  If the port is defined as COMMAND then the command will
     be issued to a DOS shell (global path will be in affect only if
     'MUIRexx' was run from a shell).  If this argument is not given
     but a command is defined then the port will be defined as the port
     for 'ARexx' (i.e.  it will be assumed that the command is an
     'ARexx' script).  Additionally, if the port is defined as INLINE
     then the command string will be treated as an inline 'ARexx' macro
     string. Note also, that the port may be defined as the port of the
     application itself.  In this manner objects within an application
     can be linked as well as to objects in other applications.

   Many examples are given in the command reference section, however,
here are a few examples illustrating the use of the INLINE port option
(note that each line within the string macros must be terminated with a
semicolon).

     callhook ID VUP ATTRS MUIA_Timer MUIV_EveryTime PORT INLINE,
         COMMAND """options results;
                    address example;
                    gauge ID VGAUG ATTRS "MUIA_Gauge_Current";
                    gauge ID VGAUG ATTRS "MUIA_Gauge_Current" result+5;"""

     callhook ID LST2 INCLUDE "LST1" DROP PORT INLINE,
         COMMAND """options results;
                    address DragnDrop;
                    line = '%s';
                    'list ID LST2 INSERT STRING' line;
                    'list ID LST1 REMOVE STRING' line;"""

     callhook ID HTML TRIG MUIA_HTMLtext_URL MUIV_EveryTime PORT INLINE,
         COMMAND """options results;
                    address DEMO;
                    object ID HTML ATTRS "MUIA_HTMLtext_URL";
                    url = import(d2c(result));
                    popasl ID SURL;
                    setvar url result;
                    popasl ID SURL CONTENT url;"""
```

1.68 MUIRexx.guide/MUI Format Sequences

```
MUI Format Sequences
********************

   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.
```

* '\033-' Disable text engine, following chars will be printed without further parsing.

* '\033u' Set the soft style to underline.

* '\033b' Set the soft style to bold.

* '\033i' Set the soft style to italic.

* '\033n' Set the soft style back to normal.

* '\033[n]' Use pen number n (2..9) as front pen. n must be a valid DrawInfo pen as specified in "intuition/screens.h".

* '\033c' Center current (and following) line(s). This sequence is only valid at the beginning of a string or after a newline character.

* '\033r' Right justify current (and following) line(s). This sequence is only valid at the beginning of a string or after a newline character.

* '\033l' Left justify current (and following) line(s). This sequence is only valid at the beginning of a string or after a newline character.

* '\033I[[s]]' Draw MUI image with specification [s] (see MUI Image Specifications).

1.69 MUIRexx.guide/MUI Image Specifications

```
MUI Image Specifications
************************

   MUI Image specifications always starts with a digit, followed by a
':', followed by some parameters.  Currently, the following things are
defined:

   * "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 (0) and MUII_Count-1 (41) identifying a preconfigured image/background.

1.70 MUIRexx.guide/MUI List Format

MUI List Format
****

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 with each line 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, BAR/S

* 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 '\033c' would make the column centered (see MUI Format Sequences).

* 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. Defaults to current entry number (0,1,...)

* BAR - Since muimaster.library V11, you can enable a vertical bar

between this and the next column by using this switch.

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.

For a dirlist object the column data, starting at column zero (0), is the file name, size, date, time, protection, and comment.

1.71 MUIRexx.guide/Attribute TAGs

Attribute TAGs
****

While the capability to specify arbitrary attributes allows for much flexibility it also can lead to unexpected results. It is recommended that before using a TAG that the MUI and AmigaOS autodocs be referenced in order to clearly understand the effect the TAG will have. All TAGs are set by specifying a TAG id and value pair. Any number of TAG pairs may be given. TAG ids should be given as hexidecimal numbers (preceded by a '0x'), although decimal numbers may also be used. Typically, TAG ids should be assigned to an 'ARexx' variable for script clarity. TAG values may be either decimal numbers, hexadecimal numbers, strings (the value will be assumed to be a string if it is not recognized as a number) or string arrays (specified by starting with a period (.) and separating each string with a comma (,). A TAG value can be forced to be interpreted as a string by prepending the string with an equal symbol (=). Note that all TAGs indicated with an 'i' flag can be set at object creation. The 's' flag indicates a TAG that can be set after object creation and the 'g' indicates a TAG that can be retrieved. To retrieve a TAG value just specify the TAG id alone. The TAG value will be returned in the 'ARexx' variable RESULT.

For commands that allow specification of attribute TAGs some example TAGs are given with the command description. However, it may be possible (even probable) that other TAGs may be usable. It is therefore highly recommended that the MUI class autodocs be consulted. These autodocs may be found in the MUI developers archive.

1.72 MUIRexx.guide/MagicUserInterface

MagicUserInterface
****

This application uses

MUI - MagicUserInterface

MUI is a system to generate and maintain graphical user interfaces. With the aid of a preferences program, the user of an application has the ability to customize the outfit according to his personal taste.

MUI is distributed as shareware. To obtain a complete package containing lots of examples and more information about registration please look for a file called "muiXXusr.lha" (XX means the latest version number) on your local bulletin boards or on public domain disks.

If you want to register directly, feel free to send

DM 30.- or US$ 20.-

to

Stefan Stuntz

Eduard-Spranger-Straße 7

80935 München

GERMANY

Support and online registration is available at

http://www.sasg.com/

1.73 MUIRexx.guide/Acknowledgements

Acknowledgements
****

* William S. Hawes for development of ARexx.

* Stefan Stuntz for MagicUserInterface (MUI).

* Tom Ekström for Iconographics

* All those responsible for the development of the Amiga.

1.74 MUIRexx.guide/History

```
History
*******

        v1.0    2/13/96 -   initial release
        v1.1    2/24/96 -   fixed enforcer hits
                            added support for menus
                            added a MUI settings command
                2/25/96 -   improved group/menu syntax checking
        v1.2i   3/10/96 -   added support for drag and drop
                3/14/96 -   added options to set MUIA attributes
                            removed unnecessary options (e.g. WEIGHT)
        v2.0    3/20/96 -   added variable storage (set and get)
                3/22/96 -   MUIA attributes are now gettable
                3/23/96 -   added support for dragging multiselected items
                            removed Cyclechain TAG (add using ATTRS)
                            removed SELECT option from check gadget
                3/24/96 -   added switch gadget
                            list objects are now dropable
                            group objects are now dropable
                3/26/96 -   stack size now set for commands
                3/28/96 -   added method command
                3/30/96 -   added object gadget
                            removed scale gadget (use object instead)
                3/31/96 -   added support for setting app attributes/methods
                            if started from WB then icon is set
                            added support for multicolumn lists
                            removed dirlist DIR and FORMAT options (use ATTRS)
                4/1/96  -   removed config and muiset commands (use method)
                4/2/96  -   added support for boopsi objects
                4/5/96  -   added support for datatype images
        v2.1    5/19/96 -   TAG strings now nonvolatile
                5/22/96 -   generalized view command
                            added aslrequest command
                5/27/96 -   improved list drag&drop behavior
                6/17/96 -   added poplist command
                6/25/96 -   added format sequences for strings
                7/20/96 -   added support for new Icon.mcc class
                            changed behavior of image and check gadgets
                8/22/96 -   added support for virtual groups
        v2.1a   9/1/96  -   minor bug fixed (popasl contents were not stored
                            in config file)
                9/4/96  -   minor bug fixed (prevented pub screen closure)
                9/5/96  -   minor update to Icon.mcc (see docs)
        v2.1b   9/7/96  -   fixed bug (enforcer hit)
                9/9/96  -   fixed bug (MUIA_Cycle_Active now works at cycle
                            object creation, thanks to Hartmut Goebel)
        v2.2    9/22/96 -   added list TITLE option
                9/25/96 -   slider, popslider and knob now return value
                            (per documentation)
                9/27/96 -   format specifiers now work for all objects
                            (I think)
                9/29/96 -   context menus for groups added
                            can now specify image for popasl/poplist objects
                            (SPEC option)
```

```
              10/1/96 -   parse routine added to replace MUI REXX command
                          line parse
              10/25/96-   fixed bug in string, popasl, poplist objects
                          (MUIA_String_MaxLen now works, thanks to Bob Sisk)
        v2.2i 11/22/96-   fixed bug in list objects (prevented sorting)
                          (thanks again to Bob Sisk)
        v3.0  12/4/96 -   added callhook command
              12/10/96-   switch objects can now have an alternate label
              12/21/96-   extended callhook to include app and drop hooks
                          removed APP DROP PRESS options (use callhook instead)
                          removed COMMAND option from group command (use ↩
                             callhook)
                          removed TRIG VAL options from object command ( ↩
                             callhook)
              12/23/96-   added REMOVE option to list object
              12/26/96-   added callhook INCLUDE/EXCLUDE to restrict drop ops
              12/29/96-   empty groups (and windows) no longer cause crashes
              1/1/97  -   groups can now be added dynamically
              1/10/97 -   fixed bug that prevented dynamic changes in popasl,
                          poplist, and list objects (discovered by Bob Sisk)
              2/13/97 -   added group POP option for creation of popup groups
              3/16/97 -   added monitor command to control debug output
              3/18/97 -   added inline command specification
              4/21/97 -   added SPEC option to button, text and switch objects
                          check and image object options changed
        v3.0a 7/24/97 -   fixed bug that prevented parsing of non-alpha ↩
           characters
                          and all characters following in strings
              7/29/97 -   added install script to install MUIRexx
```

1.75 MUIRexx.guide/Concept Index

```
Concept Index
*************
```

1.76 MUIRexx.guide/Command Index

Command Index
****

```
cycle                          cycle
dirlist                        dirlist
endgroup                       endgroup
endmenu                        endmenu
endwindow                      endwindow
gauge                          gauge
getvar                         getvar
group                          group
hide                           hide
image                          image
info                           info
item                           item
knob                           knob
label                          label
list                           list
menu                           menu
meter                          meter
method                         method
monitor                        monitor
object                         object
popasl                         popasl
poplist                        poplist
popslider                      popslider
print                          print
quit                           quit
radio                          radio
request                        request
setvar                         setvar
show                           show
slider                         slider
space                          space
string                         string
switch                         switch
text                           text
view                           view
volumelist                     volumelist
window                         window
```