TABLE OF CONTENTS

muimaster.library/--background-- muimaster.library/--background--

PURPOSE

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

muimaster.library/MUI_AddClipping muimaster.library/MUI_AddClipping

NAME

MUI_AddClipping -- restrict draw operations to a region.

SYNOPSIS

```
APTR handle = MUI_AddClipping(struct MUI_RenderInfo *mri,
    WORD left, WORD top, WORD width, WORD height);
```

FUNCTION

In certain situations it might be necessary to restrict draw operations to specific areas. Use this function to restrict future draw operations to the region defined by the supplied position and dimension. All draw operations outside this region will be skipped.

INPUTS

mri - a pointer to a struct MUI_RenderInfo

```
      left - left position of the clip region
      top - top position of the clip region
      width - width of the clip region
      height - height of the clip region

  RESULT
      handle - an abstract handle to the installed clip region or NULL in
               case of a failure.

  SEE ALSO
      MUI_AddClipRegion, MUI_RemoveClipping

muimaster.library/MUI_AddClipRegion         muimaster.library/MUI_AddClipRegion

  NAME
      MUI_AddClipRegion -- restrict draw operations to a region.

  SYNOPSIS
      APTR handle = MUI_AddClipRegion(struct MUI_RenderInfo *mri,
          struct Region *r);

  FUNCTION
      In certain situations it might be necessary to restrict draw
      operations to specific areas. Use this function to restrict future
      draw operations to the region defined by the supplied region
      structure. All draw operations outside this region will be skipped.

  INPUTS
      mri - a pointer to a struct MUI_RenderInfo
      r - a pointer to a struct Region describing the restricted region.
          MUST be created with NewRegion()!

  RESULT
      handle - an abstract handle to the installed clip region or NULL in
               case of a failure.

  SEE ALSO
      MUI_AddClipping, MUI_RemoveClipRegion

muimaster.library/MUI_AllocAslRequest     muimaster.library/MUI_AllocAslRequest

  NAME
      MUI_AllocAslRequest -- allocate an ASL requester structure.
      MUI_AllocAslRequestTags -- vararg stub for MUI_AllocAslRequest().

  SYNOPSIS
      APTR req = MUI_AllocAslRequest(ULONG type, struct TagItem *tags);

      APTR req = MUI_AllocAslRequestTags(ULONG type, ULONG tag1, ...);

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

  INPUTS
      type - type of requester to be allocated.
      tags - optional tag list specifying how to initialize the allocated
             requester.

  RESULT
      req - allocated requester structure, or NULL on failure.
```

```
  SEE ALSO
       asl.library/AllocAslRequest

muimaster.library/MUI_AslRequest              muimaster.library/MUI_AslRequest

  NAME
      MUI_AslRequest -- get input from user using an ASL requester.
      MUI_AslRequestTags -- vararg stub for MUI_AslRequest().

  SYNOPSIS
      BOOL result = MUI_AslRequest(APTR req, struct TagItem *tags);

      BOOL result = MUI_AslRequestTags(APTR req, ULONG tag1, ...);

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

  INPUTS
      req - a requester structure allocated by MUI_AllocAslRequest().
      tags - optional tag list to control features of the requester.

  RESULT
      result - boolean success value.

  SEE ALSO
       asl.library/AslRequest

muimaster.library/MUI_BeginRefresh          muimaster.library/MUI_BeginRefresh

  NAME
      MUI_BeginRefresh -- start a refresh operation in a simple refresh
          window.

  SYNOPSIS
      BOOL refresh = MUI_BeginRefresh(struct MUI_RenderInfo *mri,
          ULONG flags);

  FUNCTION
      Similar to intuition.library/BeginRefresh() this function does some
      checks and initializations before the actual refresh is initiated.
      Usually this function is called by MUI internally only. However, if
      you think that your class must assist Intuition and Layers with their
      refresh handling then this is the function to be called in favor of
      intuition.library/BeginRefresh() to let MUI take care of some further
      private stuff.
      Eventually MUI_EndRefresh() must be called after the refresh operation
      has been finished.

  INPUTS
      mri - a pointer to a struct MUI_RenderInfo
      flags - additional flags, pass 0 for future compatibility

  RESULT
      refresh - boolean value indicating whether a refresh operation was
                started.

  SEE ALSO
      MUI_EndRefresh(), intuition.library/BeginRefresh()
```

```
muimaster.library/MUI_CreateCustomClass muimaster.library/MUI_CreateCustomClass

  NAME
      MUI_CreateCustomClass -- create a public/private custom class.

  SYNOPSIS
      struct MUI_CustomClass *mcc = MUI_CreateCustomClass(
          struct Library *base, char *supername,
          struct MUI_CustomClass *supermcc, ULONG datasize, APTR dispfunc);

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

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

      MUI creates the dispatcher hook for you, you may *not* use the
      IClass->cl_Dispatcher.h_Data field! If you need custom data for your
      dispatcher, use the cl_UserData of the IClass structure or the
      mcc_UserData of the MUI_CustomClass structure.

      For public classes, MUI makes sure that A6 contains a pointer to your
      library base when your dispatcher is called. For private classes, you
      will need to keep track of A4 or similiar things the compiler may need
      yourself.

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

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

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

      datasize - size of your classes data structure.

      dispfunc - your class' dispatcher function (no hook!). The dispatcher
                 will be called with a struct IClass in A0, with your object
                 in A2 and the message in A1.

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

  SEE ALSO
      MUI_DeleteCustomClass()

muimaster.library/MUI_DeleteCustomClass muimaster.library/MUI_DeleteCustomClass

  NAME
      MUI_DeleteCustomClass -- delete a public/private custom class.
```

```
  SYNOPSIS
      BOOL result = MUI_DeleteCustomClass(struct MUI_CustomClass *mcc);

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

  INPUTS
      mcc - custom class pointer obtained from MUI_CreateCustomClass().

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

  SEE ALSO
      MUI_CreateCustomClass()

muimaster.library/MUI_DisposeObject         muimaster.library/MUI_DisposeObject

  NAME
      MUI_DisposeObject -- delete a MUI object.

  SYNOPSIS
      VOID MUI_DisposeObject(Object *object);

  FUNCTION
      Deletes a MUI object and all of it's auxiliary data. These objects are
      all created by MUI_NewObject() or NewObject(). Objects of certain
      classes "own" other objects, which will also be deleted when the
      object is passed to MUI_DisposeObject(). Read the per-class
      documentation carefully to be aware of these instances.

  INPUTS
      object - abstract pointer to a MUI object returned by MUI_NewObject().
               The pointer may be NULL, in which case this function has no
               effect.

  SEE ALSO
      MUI_NewObject(), intuition.library/SetAttrs(),
      intuition.library/GetAttr()

muimaster.library/MUI_EndRefresh               muimaster.library/MUI_EndRefresh

  NAME
      MUI_EndRefresh -- end a refresh operation in a simple refresh
          window.

  SYNOPSIS
      VOID MUI_EndRefresh(struct MUI_RenderInfo *mri, ULONG flags);

  FUNCTION
      Similar to intuition.library/EndRefresh() this function finishes a
      refresh operation initiated by MUI_BeginRefresh().

  INPUTS
      mri - a pointer to a struct MUI_RenderInfo
      flags - additional flags, pass 0 for future compatibility

  SEE ALSO
      MUI_BeginRefresh(), intuition.library/EndRefresh()
```

```
muimaster.library/MUI_Error                          muimaster.library/MUI_Error

  NAME
      MUI_Error -- return extra information from the MUI system.
          (DEPRECATED - use dos.library/IoErr() from V8+)

  SYNOPSIS
      LONG error = MUI_Error(VOID);

  FUNCTION
      This function is obsolete since MUI V8. Use dos.library/IoErr()
      instead.

  RESULT
      error - internal MUI error code.

  SEE ALSO
      dos.library/IoErr()

muimaster.library/MUI_FreeAslRequest       muimaster.library/MUI_FreeAslRequest

  NAME
      MUI_FreeAslRequest -- free file requester structure.

  SYNOPSIS
      VOID MUI_FreeAslRequest(APTR req);

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

  INPUTS
      req - a requester structure allocated by MUI_AllocAslRequest().

  SEE ALSO
       asl.library/FreeAslRequest

muimaster.library/MUI_FreeClass                 muimaster.library/MUI_FreeClass

  NAME
      MUI_FreeClass -- free class.
          (DEPRECATED - use MUI_DeleteCustomClass() from V8+)

  SYNOPSIS
      VOID MUI_FreeClass(IClass *classptr);

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

  INPUTS
      classptr - pointer to class to free.

  SEE ALSO
      MUI_CreateCustomClass(), MUI_DeleteCustomClass()

muimaster.library/MUI_GetClass                   muimaster.library/MUI_GetClass

  NAME
      MUI_GetClass -- get a pointer to a MUI class.
          (DEPRECATED - use MUI_CreateCustomClass() from V8+)
```

```
  SYNOPSIS
      struct IClass *classptr = MUI_GetClass(char *classID);

  FUNCTION
      This function is obsolete since MUI V8. Use MUI_CreateCustomClass()
      instead.

  INPUTS
      classID - name of public class to get.

  SEE ALSO
      MUI_CreateCustomClass(), MUI_DeleteCustomClass()

muimaster.library/MUI_Hide                          muimaster.library/MUI_Hide

  NAME
      MUI_Hide -- hide yourself.

  SYNOPSIS
      ULONG rc = MUI_Hide(Object *obj);

  FUNCTION
      With MUI_Hide(), an object tells itself to hide, e.g. when some
      internal attributes were changed which require a hidden/visible
      transition of the object. Calling MUI_Hide() is only legal within a
      custom class dispatcher, using this function within an application's
      main part is invalid!
      This function effectively invokes the object's MUIM_Hide method.

  INPUTS
      obj - pointer to yourself as an object.

  RESULT
      rc - return value of the MUIM_Hide method

  SEE ALSO
      MUI_Hide, Area.mui/MUIM_Hide, Area.mui/MUIM_Show

muimaster.library/MUI_Layout                      muimaster.library/MUI_Layout

  NAME
      MUI_LayoutObj -- layout yourself.

  SYNOPSIS
      BOOL rc = MUI_LayoutObj(Object *obj, LONG left, LONG top, LONG width,
          LONG height, ULONG flags);

  FUNCTION
      With MUI_LayoutObj(), an object is told its final position and
      dimension within the window. Calling MUI_LayoutObj() is only legal
      within a custom class dispatcher, using this function within an
      application's main part is invalid!
      This function effectively invokes the object's MUIM_Layout method or
      a possible layout hook.

      The placement of the object is done absolute within the object's
      window.

  INPUTS
      obj - pointer to yourself as an object.
      left - wanted left position for the object
```

```
      top - wanted top position for the object
      width - wanted width for the object
      height - wanted height for the object
      flags - additional flags, pass 0 for future compatibility

  RESULT
      rc - boolean return value of the MUIM_Layout method. Be prepared for
           possible failure.

  SEE ALSO
      MUI_Layout

muimaster.library/MUI_MakeObjectA            muimaster.library/MUI_MakeObjectA

  NAME
      MUI_MakeObjectA -- create an object from the builtin object
                         collection.
      MUI_MakeObject -- varargs stub for MUI_MakeObjectA().

  SYNOPSIS
      Object *object = MUI_MakeObjectA(ULONG type, ULONG *params);

      Object *object = MUI_MakeObject(ULONG type, ...);

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

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

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

  INPUTS
      type - type of object to be created
      params - additional type specific parameters

  RESULT
      object - pointer to a newly created object or NULL on failure.

  SEE ALSO
      MUI_CreateCustomClass(), MUI_DeleteCustomClass()

muimaster.library/MUI_NewObjectA              muimaster.library/MUI_NewObjectA

  NAME
      MUI_NewObjectA -- create an object from a class.
      MUI_NewObject -- varargs stub for MUI_NewObjectA().

  SYNOPSIS
      Object *object = MUI_NewObjectA(CONST_STRPTR classID,
          struct TagItem *tags);

      Object *object = MUI_NewObject(CONST_STRPTR classID, ULONG tag1, ...);

  FUNCTION
      This is the general method of creating objects from MUI classes. You
```

```
      specify a class by its ID string. If the class is not already in
      memory or built into muimaster.library, it will be loaded using
      OpenLibrary("mui/%s", 0).

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

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

      tags - pointer to array of TagItems containing attribute/value pairs
             to be applied to the object being created.

  RESULT
      A MUI object, which may be used in different contexts such as an
      application, window or gadget, and may be manipulated by generic
      functions. You eventually free the object using MUI_DisposeObject().
      NULL indicates failure.

  SEE ALSO
      MUI_DisposeObject(), intuition.library/SetAttrs(),
      intuition.library/GetAttr()

muimaster.library/MUI_ObtainPen                muimaster.library/MUI_ObtainPen

  NAME
      MUI_ObtainPen -- obtain a drawing pen.

  SYNOPSIS
      LONG pen = MUI_ObtainPen(struct MUI_RenderInfo *mri,
          const struct MUI_PenSpec *spec, ULONG flags);

  FUNCTION
      Whenever your application needs custom drawing pens, you should allow
      your user to adjust them with a Poppen class. The result from this
      Poppen class is a struct MUI_PenSpec which you can save somewhere in
      your preferences and use together with MUI_ObtainPen(),
      MUI_ReleasePen() and the MUIPEN() macro to transform the spec into a
      pen number useful for graphics.library/SetAPen().

      MUI knows 5 different kinds of pen specs:
      1. system pens, i.e. "s0". This is equivalent to using the pens from
         muiRenderInfo(obj)->mri_DrawInfo->dri_Pens directly. Make sure to
         use pen numbers between and mri_DrawInfo->dri_NumPens-1 only.
         Otherwise the default BACKGROUND pen will be returned.
      2. MUI pens, i.e. "m5". This is equivalent to using the pens from
         muiRenderInfo(obj)->mri_Pens directly. Make sure to use pen numbers
         between and MPEN_COUNT-1 only. Otherwise the MPEN_BACKGROUND pen
         will be returned.
      3. direct palette entries, i.e. "p42". This will use pen #42 from the
         screen's current palette. This kind of spec should only be used if
         you really know which color you get as no allocation is done.
         Therefore the usage of this kind of spec is not recommended.
      4. direct RGB values, i.e. "rRRGGBB" or "rRRRRRRRR,GGGGGGGGG,BBBBBBBB".
         This kind of spec will let MUI allocate a pen with the given RGB
         values (either 3x8 bit or 3x24 bits).
      5. an empty string. This will return the default text pen/color.
```

```
  INPUTS
      mri - a pointer to a valid MUI_RenderInfo structure
      spec - a string describing the pen to be obtained
      flags - current unused flags, set to 0 for future compatibility

  RESULT
      pen - number of the obtained pen or -1 on failure.

  EXAMPLE
      See demo program Class2.c

  SEE ALSO
      MUI_ReleasePen, Poppen.mui

muimaster.library/MUI_Redraw                        muimaster.library/MUI_Redraw

  NAME
      MUI_Redraw -- redraw yourself.

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

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

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

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

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

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

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

  INPUTS
      obj - a pointer to the object to be redrawn
      flag - MADF_DRAWOBJECT or MADF_DRAWUPDATE.
             The flag given here affects the objects flags when MUI calls
             the MUIM_Draw method. There are several caveats when
             implementing MUIM_Draw, see the developer documentation for
             details.
```

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

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

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

          case MYATTR_PEN_1:
            data->pen2 = tag->ti_Data;        // set the new value
            data->mark = 2;                   // set internal marker
            MUI_Redraw(obj, MADF_DRAWUPDATE); // update ourselves
          break;
        }
      }

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

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

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

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

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

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

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

      return 0;
```

```
      }

  SEE ALSO
      Area.mui/MUIM_Draw

muimaster.library/MUI_RejectIDCMP            muimaster.library/MUI_RejectIDCMP

  NAME
      MUI_RejectIDCMP -- reject previously requested input events.
          (DEPRECATED - use MUIM_HandleEvent instead)

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

  FUNCTION
      OBSOLETE! Please use event handlers for your classes input needs.

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

      Critical flags such as IDCMP_MOUSEMOVE and IDCMP_INTUITICKS should be
      rejected as soon as possible. See MUI_RequestIDCMP() for details.

  INPUTS
      obj - pointer to yourself as an object.
      flags - one or more IDCMP_XXXX flags.

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

  SEE ALSO
      MUI_RequestIDCMP(), Area.mui/MUIM_HandleEvent,
      Window.mui/MUIM_Window_AddEventHandler,
      Window.mui/MUIM_Window_RemEventHandler

muimaster.library/MUI_ReleasePen              muimaster.library/MUI_ReleasePen

  NAME
      MUI_ReleasePen -- release a drawing pen.

  SYNOPSIS
      VOID MUI_ReleasePen(struct MUI_RenderInfo *mri, ULONG pen);

  FUNCTION
      Release a pen obtained with MUI_ObtainPen(). Usually placed in the
      Cleanup method of custom classes.

  NOTE
      Do *not* use the MUIPEN() maros when releasing pens!

  INPUTS
      mri - a pointer to a valid MUI_RenderInfo structure
      pen - the pen number to be released

  EXAMPLE
      See demo program Class2.c
```

```
  SEE ALSO
      MUI_ObtainPen, Poppen.mui

muimaster.library/MUI_RemoveClipping       muimaster.library/MUI_RemoveClipping

  NAME
      MUI_RemoveClipping -- remove a clip region.

  SYNOPSIS
      VOID MUI_RemoveClipping(struct MUI_RenderInfo *mri, APTR handle);

  FUNCTION
      Removed a formerly added clip region. All further draw operations will
      be perfomed without clipping again.

  INPUTS
      mri - a pointer to a struct MUI_RenderInfo
      handle - a handle returned by MUIA_AddClipping()

  SEE ALSO
      MUI_AddClipping

muimaster.library/MUI_RemoveClipRegion   muimaster.library/MUI_RemoveClipRegion

  NAME
      MUI_RemoveClipRegion -- remove a clip region.

  SYNOPSIS
      VOID MUI_RemoveClipRegion(struct MUI_RenderInfo *mri, APTR handle);

  FUNCTION
      Removed a formerly added clip region. All further draw operations will
      be perfomed without clipping again.
      The region pointer passed to MUI_AddClipRegion() will be freed by this
      function! Do not free it yourself!

  INPUTS
      mri - a pointer to a struct MUI_RenderInfo
      handle - a handle returned by MUI_AddClipRegion()

  SEE ALSO
      MUI_AddClipRegion

muimaster.library/MUI_RequestA                    muimaster.library/MUI_RequestA

  NAME
      MUI_RequestA -- pop up a MUI requester.
      MUI_Request -- vararg stub for MUI_Request().

  SYNOPSIS
      LONG gad = MUI_Request(Object *app, Object *win, ULONG flags,
          CONST_STRPTR title, CONST_STRPTR gadgets, CONST_STRPTR format,
          APTR params);

      LONG gad = MUI_Request(Object *app, Object *win, ULONG flags,
          CONST_STRPTR title, CONST_STRPTR gadgets, CONST_STRPTR format,
          ...);

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

```
  INPUTS
      app - The application object. If you leave this NULL, MUI_RequestA()
            will fall back to a standard system requester.

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

      flags - Define an image type to be displayed left of the text. See
              mui.h for the available types. If the default type is used
              then depending on the number of gadgets either an info sign
              (single gadget) or a question sign (two or more gadgets) will
              be displayed.
              Defaults to MUIV_Requester_Type_Default.
              NOTE: This feature is available since MUI 4.0-2015R3. All
                    previous releases will simply ignore it.
              NOTE: As an alternative to the image type the flags variable
              can be passed the flag MUIREQ_XYPOS ORed with an absolute
              position for the requester.
              For example "MUIREQ_XYPOS|xpos<<16|ypos" will open the
              requester with a left position of "xpos" and a top position of
              "ypos". In this case NO custom image can be used, but only the
              default images.

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

      gadgets - Pointer to a string containing the possible answers. The
                format looks like "_Save|_Use|_Test|_Cancel". If you precede
                an entry with a '*', this answer will become the active
                object. Pressing <Return> will terminate the requester with
                this response. A '_' character indicates the keyboard
                shortcut for this response.

      format - A printf-style formatting string. This will be parsed using
               exec/RawDoFmt(), thus all the usual rules apply here.

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

  RESULT
      gad - 0, 1, ..., N - Successive ID values, for the gadgets you specify
                           for the requester.
                           NOTE: The numbering from left to right is
                           actually: 1, 2, ..., N, 0. In case of a problem
                           (e.g. out of memory), this function returns
                           FALSE.

  SEE ALSO
       Area.mui/MUIA_Text_Contents, exec.library/RawDoFmt

muimaster.library/MUI_RequestIDCMP           muimaster.library/MUI_RequestIDCMP

  NAME
      MUI_RequestIDCMP -- request input events for your custom class.
          (DEPRECATED - use MUIM_HandleEvent instead)

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

  FUNCTION
      OBSOLETE! Please use event handlers for your classes input needs.
```

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

      Whenever an input event you requested arrives at your parent window's
      message port, your object will receive a MUIM_HandleInput method.

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

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

  INPUTS
      obj - pointer to yourself as an object.
      flags - one or more IDCMP_XXXX flags.

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

        // do some setup here...
        ...

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

        return TRUE;
      }

  SEE ALSO
      MUI_RejectIDCMP(), Area.mui/MUIM_HandleEvent,
      Window.mui/MUIM_Window_AddEventHandler,
      Window.mui/MUIM_Window_RemEventHandler

muimaster.library/MUI_SetError                  muimaster.library/MUI_SetError

  NAME
      MUI_SetError -- set an error value.
          (DEPRECATED - use dos.library/SetIoErr() from V8+)

  SYNOPSIS
      VOID MUI_SetError(LONG error);

  FUNCTION
      This function is obsolete since MUI V8. Use dos.library/SetIoErr()
      instead.

  INPUTS
      error - error code to be set.

  SEE ALSO
```

```
      dos.library/SetIoErr()

muimaster.library/MUI_Show                              muimaster.library/MUI_Show

  NAME
      MUI_Show -- show yourself.

  SYNOPSIS
      ULONG rc = MUI_Show(Object *obj);

  FUNCTION
      With MUI_Show(), an object tells itself to show, e.g. when some
      internal attributes were changed which require a hidden/visible
      transition of the object. Calling MUI_Show() is only legal within a
      custom class dispatcher, using this function within an application's
      main part is invalid!
      This function effectively invokes the object's MUIM_Show method.

  INPUTS
      obj - pointer to yourself as an object.

  RESULT
      rc - return value of the MUIM_Show method

  SEE ALSO
      MUI_Hide, Area.mui/MUIM_Hide, Area.mui/MUIM_Show

muimaster.library/MUI_Layout                          muimaster.library/MUI_Layout

  NAME
      MUI_Layout -- layout yourself.

  SYNOPSIS
      BOOL rc = MUI_Layout(Object *obj, LONG left, LONG top, LONG width,
          LONG height, ULONG flags);

  FUNCTION
      With MUI_Layout(), an object is told its final position and dimension
      within the window with respect to the object's parent object's
      position. Calling MUI_Layout() is only legal within a custom class
      dispatcher, using this function within an application's main part is
      invalid!
      This function effectively invokes the object's MUIM_Layout method or
      a possible layout hook.

      The supplied left/top position will be added to the object's parent
      object's position to archive a placement relative to the parent
      object.

  INPUTS
      obj - pointer to yourself as an object.
      left - wanted left position for the object
      top - wanted top position for the object
      width - wanted width for the object
      height - wanted height for the object
      flags - additional flags, pass 0 for future compatibility

  RESULT
      rc - boolean return value of the MUIM_Layout method. Be prepared for
           possible failure.

  SEE ALSO
```

```
MUI_LayoutObj
```