079d5630-0

Deryk B Robosson

COLLABORATORS

	TITLE : 079d5630-0	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Deryk B Robosson	November 23, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

079d5630-0

1.1 AFReqTools

```
                        **********************************

                          ReqTools C++ Object

                          The ReqTools.library Wrapper.

                          AFrame Version 1.0
                          ReqTools Object Version 1.0

                          (c) 1995,1996 Jeffry A Worth
                                        Deryk B Robosson

                        **********************************

TABLE OF CONTENTS

  ReqTools~Object~Information

  Methods:
    AFReqTools()
    ~AFReqTools

    RTCreate
    RTEZRequest
    RTEZRequestA
    RTScreenMode
    RTScreenModeA
    RTFileRequest
    RTFileRequestA
    RTFilesRequest
    RTFilesRequestA
    RTDirRequest
    RTVolumeRequest
    RTFontRequest
    RTFontRequestA
    RTPaletteRequest
    RTPaletteRequestA
    RTScreenToFront
```

```
    RTSetWaitPointer
    RTLockWindow
    RTUnlockWindow
    RTGetLong
    RTGetLongA
    RTGetString
    RTGetStringA
    RTGetVScreenSize

    Misc~Members

    Structs

    Hooks

    Includes
    Source
    History
    Distribution
```

1.2 AFReqTools

```
AFReqTools/AFReqTools

  NAME AFReqTools()

  DESCRIPTION
    Default class constructor.  Modify this only if you wish each class to have
    the modifications

  INPUTS
    none

  RESULT
    none

  BUGS
    none known

  SEE ALSO
    ~AFReqTools
```

1.3 ~AFReqTools

```
AFReqTools/~AFReqTools

  NAME ~AFReqTools()

  DESCRIPTION
    Default class destructor.  Modify this only if you wish each class to have
    the modifications
```

```
  INPUTS
    none

  RESULT
    none

  BUGS
    none known

  SEE ALSO
    AFReqTools
```

1.4 RTCreate

```
AFReqTools/RTCreate

  NAME RTCreate()

    ret = RTCreate();

  DESCRIPTION
    The default Create function tests to see if reqtools.library is open, if not
    then will open the library.

  NOTE
    If you wish to use this class under 1.2 or 1.3 kickstarts you will have to
    override this function to open the correct version of your library

  INPUTS
    none

  RESULT
    TRUE if success
    FALSE if failure to open correct version of library

  BUGS
    none known

  SEE ALSO
```

1.5 RTScreenMode

```
AFReqTools/RTScrenMode

  NAME RTScreenMode()

    ret = RTScreenMode();

    ret = RTScreenModeA(ULONG *)

  DESCRIPTION
```

```
    *IMPORTANT* THIS REQUESTER IS ONLY AVAILABLE FROM KICKSTART 2.0 ONWARDS!
                The 1.3 version of ReqTools also contains the screenmode
                requester, but unless you are running 2.0 or higher it will
                not come up.  So what you essentially have to do is NOT
                call RTScreenModeA() if your program is running on
                a machine with Kickstart 1.2/1.3.  You can safely call
                RTScreenModeA() if you are running on a 2.0 machine,
                even if the user has installed the 1.3 version of ReqTools.

    Get a screen mode from the user.

    The user will be able to pick a screen mode by name, enter the size and
    the number of colors (bitplane depth).

    rtScreenModeRequestA() will call the appropriate 2.0 functions to get all
    the mode's information.  If no name has been assigned to the mode one
    will be constructed automatically.

  INPUTS
    If using RTScreenModeA it expects to see a taglist of items for customizing
    the requester.

    EXAMPLE: ULONG taglist[] = { ITEM1, ITEM2, ITEM3, ..., TAG_END };

  NOTE
    See reqtools developer documentation for descriptions of tag items which
    can be used with RTScreenModeA

  RESULT
    TRUE if success
    FALSE if failure to open requester

  BUGS
    none known

  SEE ALSO
    m_screenmoderequester
```

1.6 RTFileRequest

```
AFReqTools/RTFileRequest

  NAME RTFileRequest()

    ret = RTFileRequest();

    ret = RTFileRequestA(ULONG *)

  DESCRIPTION
    Get a filename from the user.

    'filename' should point to an array of at least 108 chars.  The filename
    already in 'filename' will be displayed in the requester when it comes
    up.  When the requester returns 'filename' will probably have changed.
```

```
    Using certain tags may result in the calling of a caller-supplied hook.

    The hook will be called with A0 holding the address of your hook
    structure (you may use the h_Data field to your own liking), A2 a pointer
    to the requester structure calling the hook ('req') and A1 a pointer to
    an object.  The object is variable and depends on what your hook is for.


  INPUTS
    If using RTFileRequestA it expects to see a taglist of items for customizing
    the requester.

    EXAMPLE: ULONG taglist[] = { ITEM1, ITEM2, ITEM3, ..., TAG_END };

  NOTE
    See reqtools developer documentation for descriptions of tag items which
    can be used with RTFileRequestA

  RESULT
    TRUE if success
    FALSE if failure to open requester

  BUGS
    none known

  SEE ALSO
    RTFilesRequest
    m_filerequester
    m_filterhook
    m_volume_filterhook
    Hooks
```

1.7 RTFilesRequest

```
AFReqTools/RTFilesRequest

  NAME RTFilesRequest()

    ret = RTFilesRequest();

    ret = RTFilesRequestA(ULONG *)

  DESCRIPTION
    Get filename(s) from the user.

    'filename' should point to an array of at least 108 chars.  The filename
    already in 'filename' will be displayed in the requester when it comes
    up.  When the requester returns 'filename' will probably have changed.

    Using certain tags may result in the calling of a caller-supplied hook.

    The hook will be called with A0 holding the address of your hook
    structure (you may use the h_Data field to your own liking), A2 a pointer
    to the requester structure calling the hook ('req') and A1 a pointer to
    an object.  The object is variable and depends on what your hook is for.
```

```
  INPUTS
    If using RTFileRequestA it expects to see a taglist of items for customizing
    the requester.

    EXAMPLE: ULONG taglist[] = { ITEM1, ITEM2, ITEM3, ..., TAG_END };

  NOTE
    See reqtools developer documentation for descriptions of tag items which
    can be used with RTFileRequestA

  RESULT
    TRUE if success
    FALSE if failure to open requester

  BUGS
    none known

  SEE ALSO
    RTFileRequest
    m_filerequester
    m_filterhook
    m_volume_filterhook
    Hooks
```

1.8 RTDirRequest

```
AFReqTools/RTDirRequest

  NAME RTDirRequest()

    ret = RTDirRequest();

  DESCRIPTION
    Get a directory from the user.  What follows is an example
    of directory extraction.

    EXAMPLE:  AFReqTools rt;
              printf("Directory: %s\n",rt.m_filerequester->Dir);

  NOTE
    The hook function is not implemented at this time

  INPUTS
    none

  RESULT
    TRUE if success
    FALSE if failure to open requester

  BUGS
    none known

  SEE ALSO
```

```
    m_filerequester
    m_volume_filterhook
    Hooks
```

1.9 RTVolumeRequest

```
AFReqTools/RTVolumeRequest

  NAME RTDirRequest()

    ret = RTDirRequest();

  DESCRIPTION
    Get a volume from the user.  What follows is an example
    of volume extraction.

    EXAMPLE:  AFReqTools rt;
              printf("Volume: %s\n",rt.m_filerequester->Dir);

  INPUTS
    none

  RESULT
    TRUE if success
    FALSE if failure to open requester

  BUGS
    none known

  SEE ALSO
    m_filerequester
    m_volume_filterhook
    Hooks
```

1.10 RTFontRequest

```
AFReqTools/RTFontRequest

  NAME RTFontRequest()

    ret = RTFontRequest();

    ret = RTFontRequest(ULONG *);

  DESCRIPTION
    Get a font from the user.  What follows is an example
    of font extraction.

    EXAMPLE:  AFReqTools rt;
              printf("Fontname: %s\nSize: %ld",
                      rt.m_fontrequester->Attr.ta_Name,
                      rt.m_fontrequester->Attr.ta_YSize);
```

```
  NOTE
    See reqtools developer documentation for descriptions of tag items which
    can be used with RTFileRequestA

  INPUTS
    If using RTFileRequestA it expects to see a taglist of items for customizing
    the requester.

    EXAMPLE: ULONG taglist[] = { ITEM1, ITEM2, ITEM3, ..., TAG_END };

  RESULT
    TRUE if success
    FALSE if failure to open requester

  BUGS
    none known

  SEE ALSO
    m_fontrequester
    m_font_filterhook
    Hooks
```

1.11 RTPaletteRequest

```
AFReqTools/RTPaletteRequest

  NAME RTPaletteRequest()

    ret = RTPaletteRequest();

    ret = RTPaletteRequest(ULONG *);

  DESCRIPTION
    Get a color from the user.

  NOTE
    See reqtools developer documentation for descriptions of tag items which
    can be used with RTFileRequestA

  INPUTS
    If using RTFileRequestA it expects to see a taglist of items for customizing
    the requester.

    EXAMPLE: ULONG taglist[] = { ITEM1, ITEM2, ITEM3, ..., TAG_END };

  RESULT
    TRUE if success
    FALSE if failure to open requester

    The member color is set to the selected color

  BUGS
    none known
```

```
  SEE ALSO
    rtPaletteRequest()
    rtPaletteRequestA()
    color
```

1.12 RTScreenToFront

```
AFReqTools/RTScreenToFront

  NAME RTScreenToFront()

    RTScreenToFront(struct Screen *);

  DESCRIPTION
    Brings the specified screen to the front of the display, but only after
    checking it is still in the list of currently open screens.

    This function can be used to bring a screen back to the front of the
    display after bringing another screen to the front.  If the first screen
    closed while you where busy it is harmless to call this function, unlike
    calling the normal ScreenToFront().

  NOTE
    This function is for the advanced ReqTools user.

  INPUTS
    screen - pointer to the screen

  RESULT
    none

  BUGS
    none known

  SEE ALSO
   intuition.library/ScreenToFront()
```

1.13 RTSetWaitPointer

```
AFReqTools/RTSetWaitPointer

  NAME RTSetWaitPointer()

    RTSetWaitPointer(struct Window *);

  DESCRIPTION
    Change the window's pointer image to that of a wait pointer.  Call this
    function whenever your program will be busy doing something for a lengthy
    period of time.

    It is recommended you call this function before calling any of the
    requester functions.  This way if the user clicks in your window he will
```

```
    know he must respond to the requester before doing anything else.  Also
    see the RT_WaitPointer tag for an automatic way of setting the wait
    pointer.  If you are using ReqTools V38+ check out the RT_LockWindow tag!

  NOTE
    The wait pointer will look exactly like the standard Workbench 2.0
    wait pointer.  In combination with PointerX, ClockTick or LacePointer
    the handle will turn.


  INPUTS
    window - pointer to the window to receive the wait pointer

  RESULT
    none

  BUGS
    none known

  SEE ALSO
```

1.14 RTLockWindow

```
AFReqTools/RTLockWindow

  NAME RTLockWindow()

    m_windowlock = RTLockWindow(struct Window *);

  DESCRIPTION
    Lock a window so it will no longer accept any user input.  The only
    functions left to the user are depth arrangement and window dragging.
    All gadgets will be un-selectable and the window can not be resized.
    It will also get the standard wait pointer set.  The pointer at the
    time of locking will be restored when the window is unlocked (this
    will *not* happen on Kickstart V39 or higher!).

    You may nest calls to RTLockWindow() and RTUnlockWindow().  Make sure
    you unlock the window in the correct (opposite) order.

  NOTE
    The wait pointer will look exactly like the standard Workbench 2.0
    wait pointer.  In combination with PointerX, ClockTick or LacePointer
    the handle will turn.

  INPUTS
    windowlock - a pointer to a (private) window lock.  You must pass this
                 to RTUnlockWindow() to unlock the window again.
                 Never mind if this is NULL.  This means there was not enough
                 memory and the window will not be locked.  There is no
                 sense in reporting this, just carry on and pass the NULL
                 window lock to RTUnlockWindow().

  RESULT
    none
```

```
  BUGS
    none known

  SEE ALSO
    RTUnlockWindow
    m_windowlock
```

1.15 RTUnlockWindow

```
AFReqTools/RTUnlockWindow

  NAME RTUnlockWindow()

    RTUnlockWindow(window, windowlock);

  DESCRIPTION
    Unlock a window previously locked with RTLockWindow().  The window will
    once again accept user input and will get its original mouse pointer
    back (default or custom).

    Under Kickstart V39 or higher the original window pointer will not be
    restored if it was set using SetWindowPointer().  You will have to
    restore the pointer yourself in this case.

  INPUTS
    window     - pointer to the window to be unlocked.
    windowlock - the windowlock pointer returned by rtLockWindow(), may
                 be NULL.

  RESULT
    none

  BUGS
    none known

  SEE ALSO
    RTLockWindow
    m_windowlock
```

1.16 RTGetLong

```
AFReqTools/RTGetLong

  NAME RTGetLong()

    ULONG ret = RTGetLong(ULONG var, char *title, struct rtReqInfo *reqinfo);

    ULONG ret = RTGetLong(ULONG var, char *title, struct rtReqInfo *reqinfo,
                          ULONG *taglist);

  DESCRIPTION
```

```
    Puts up a requester to get a signed long (32-bit) number from the user.

    'reqinfo' can be used to customize the requester.  For greater control
    use the tags listed below.  The advantage of the rtReqInfo structure is
    that it is global, where tags have to be specified each function call.
    See libraries/reqtools.[hi] for a description of the rtReqInfo structure.

  NOTE
    'var' will NOT change if the requester is aborted.

    See reqtools developer documentation for descriptions of tag items which
    can be used with RTFileRequestA

  INPUTS
    &var - address of long (32 bit!) variable to hold result.
    title    - pointer to null terminated title of requester window.
    reqinfo  - pointer to a rtReqInfo structure allocated with
               rtAllocRequest() or NULL.
    taglist  - pointer to a TagItem array.

  RESULT
    ret - TRUE if user entered a number, FALSE if not. If one of your idcmp
          flags caused the requester to end 'ret' will hold this flag.
          If you used the RTGL_GadFmt tag the return code will hold the
          value of the response as with rtEZRequestA().

  BUGS
    none known

  SEE ALSO
    m_reqinfo
```

1.17 RTGetString

```
AFReqTools/RTGetString

  NAME RTGetString()

    ULONG ret = RTGetString(UBYTE *buffer, ULONG maxchars, char *title,
                            struct rtReqInfo *reqinfo);

    ULONG ret = RTGetString(ULONG *buffer, ULONG maxchars, char *title,
                            struct rtReqInfo *reqinfo, ULONG *taglist);

  DESCRIPTION
    Puts up a string requester to get a line of text from the user.
    The string present in 'buffer' upon entry will be displayed, ready to
    be edited.

    'reqinfo' can be used to customize the requester.  For greater control
    use the tags listed below.  The advantage of the rtReqInfo structure is
    that it is global, where tags have to be specified each function call.
    See libraries/reqtools.[hi] for a description of the rtReqInfo structure.

  NOTE
```

```
    The contents of the buffer will NOT change if the requester is aborted.

    See reqtools developer documentation for descriptions of tag items which
    can be used with RTFileRequestA

  INPUTS
    buffer   - pointer to buffer to hold characters entered.
    maxchars - maximum number of characters that fit in buffer (EX-cluding
               the 0 to terminate the string !).
    title    - pointer to null terminated title of requester window.
    reqinfo  - pointer to a rtReqInfo structure allocated with
               rtAllocRequest() or NULL.
    taglist  - pointer to a TagItem array.

  RESULT
    ret - TRUE if user entered a number, FALSE if not. If one of your idcmp
          flags caused the requester to end 'ret' will hold this flag.
          If you used the RTGL_GadFmt tag the return code will hold the
          value of the response as with rtEZRequestA().

  BUGS
    none known

  SEE ALSO
    AFReqTools buffer member
```

1.18 RTGetVScreenSize

```
AFReqTools/RTGetVScreenSize

  NAME RTGetVScreenSize()

    ULONG spacing = RTGetVScreenSize(struct Screen *screen, ULONG *widthptr,
                                     ULONG *heightptr);

  DESCRIPTION
    Use this function to get the size of the visible portion of a screen.

    The value returned by RTGetVScreenSize() can be used for vertical
    spacing.  It will be larger for interlaced and productivity screens.
    Using this number for spacing will assure your requester will look
    good on an interlaced and a non-interlaced screen.

    Current return codes are 2 for non-interlaced and 4 for interlaced.
    These values may change in the future, don't depend on them too much.
    They will in any case remain of the same magnitude.

  NOTE
    This function is for the advanced ReqTools user.

  INPUTS
    screen    - pointer to the screen.
    widthptr  - address of an ULONG variable to hold the width.
    heightptr - address of an ULONG variable to hold the height.
```

```
  RESULT
    spacing - vertical spacing for the screen

  BUGS
    none known

  SEE ALSO
```

1.19 Misc Members

```
AFReqTools/Misc Members

 m_reqinfo
 m_filerequester
 m_fontrequester
 m_screenmoderequester
 m_filelist
 m_tempfilelist

 m_filterhook
 m_font_filterhook
 m_volume_filterhook

 m_windowlock
 color
 buffer
 filename
 longvar
```

1.20 Structs

```
AFReqTools/Structs

    NAME rtReqInfo, rtFileRequester, rtFontRequester, rtScreenModeRequester
         rtFileList, Hook

    DESCRIPTION
      Miscellaneous structures used by the AFReqTools class object

    INPUTS
      none

    RESULT
      none

    BUGS
      none known

    SEE ALSO
      struct~rtReqInfo
      struct~rtFileRequester
      struct~rtFontRequester
```

```
  struct~rtScreenModeRequester
  struct~rtFileList
  <utility/hooks.h>
```

1.21 Hooks

```
AFReqTools/Hooks

    NAME BOOL __asm __saveds file_filterfunc (
              register __a0 struct Hook *, register __a2 struct rtFileRequester *,
              register __a1 struct FileInfoBlock *
         );

    NAME BOOL __asm __saveds font_filterfunc (
              register __a0 struct Hook *, register __a2 struct rtFontRequester *,
              register __a1 struct TextAttr *
         );

    NAME BOOL __asm __saveds vol_filterfunc (
              register __a0 struct Hook *, register __a2 struct rtFileRequester *,
              register __a1 struct rtVolumeEntry *
         );


    DESCRIPTION
      Used to filter unwanted items in a requester so the user does no
      have them as an option.

    INPUTS
      hook      - struct Hook pointer
      rtFileRequester, rtFontRequester
                - reqtools requester struct pointer
      rtVolumeEntry, TextAttr, FileInfoBlock
                - appropriate struct for information comparison

    RESULT
      BOOL      - TRUE if item examined is accepted, FALSE if not

    BUGS
      none known

    SEE ALSO
```

1.22 Includes

```
  ////////////////////////////////////////////////////////////////////////////
  // ReqTools.hpp - AFrame v1.0 © 1996 Synthetic Input
  //
  // ReqTools C++ Object utilizing reqtools.library version 38.1296
  // by Nico François.  ReqTools is © Nico François
  //
  //
```

```
// Deryk B Robosson
// Jeffry A Worth
// January 20, 1996
////////////////////////////////////////////////////////////////////////////

#ifndef __AFREQTOOLS_HPP__
#define __AFREQTOOLS_HPP__

////////////////////////////////////////////////////////////////////////////
// INCLUDES
#include "AFrame:include/AFrame.hpp"
#include "AFrame:include/Object.hpp"
#include <exec/types.h>
#include <exec/memory.h>
#include <exec/execbase.h>
#include <intuition/intuition.h>
#include <utility/tagitem.h>
#include <libraries/dos.h>
#include <proto/exec.h>
#include <proto/dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <libraries/reqtools.h>
#include <proto/reqtools.h>

#define REG register

BOOL __asm __saveds file_filterfunc (
    REG __a0 struct Hook *, REG __a2 struct rtFileRequester *,
    REG __a1 struct FileInfoBlock *
);

BOOL __asm __saveds font_filterfunc (
    REG __a0 struct Hook *, REG __a2 struct rtFontRequester *,
    REG __a1 struct TextAttr *
);

BOOL __asm __saveds vol_filterfunc (
    REG __a0 struct Hook *, REG __a2 struct rtFileRequester *,
    REG __a1 struct rtVolumeEntry *
);

////////////////////////////////////////////////////////////////////////////
// Req_Tools Class
class AFReqTools : public AFObject
{
public:

    AFReqTools();
    ~AFReqTools();

    virtual void DestroyObject();
    virtual char *ObjectType() { return "ReqTools"; };

// Methods
```

```
    virtual BOOL RTCreate();
    virtual BOOL RTEZRequest(char *, char *);
    virtual BOOL RTEZRequestA(char *, char *, struct rtReqInfo *, ULONG *);
    virtual BOOL RTScreenMode();
    virtual BOOL RTScreenModeA(ULONG *);
    virtual BOOL RTFileRequest();
    virtual BOOL RTFileRequestA(ULONG *);
    virtual BOOL RTFilesRequest();
    virtual BOOL RTFilesRequestA(ULONG *);
    virtual BOOL RTDirRequest();
    virtual BOOL RTVolumeRequest();
    virtual BOOL RTFontRequest();
    virtual BOOL RTFontRequestA(ULONG *);
    virtual BOOL RTPaletteRequest();
    virtual BOOL RTPaletteRequestA(ULONG *);
    virtual void RTScreenToFront(struct Screen *);
    virtual void RTSetWaitPointer(struct Window *);
    virtual BOOL RTLockWindow(struct Window *);
    virtual void RTUnlockWindow(struct Window *,APTR);
    virtual ULONG RTGetLong(char *, struct rtReqInfo *);
    virtual ULONG RTGetLongA(char *, struct rtReqInfo *, ULONG *);
    virtual ULONG RTGetString(UBYTE *, ULONG, char *, struct rtReqInfo *);
    virtual ULONG RTGetStringA(UBYTE *, ULONG, char *, struct rtReqInfo *, ULONG ↩
        *);
    virtual ULONG RTGetVScreenSize(struct Screen *, ULONG *, ULONG *);

    struct rtReqInfo *m_reqinfo;
    struct rtFileRequester *m_filerequester;
    struct rtFontRequester *m_fontrequester;
    struct rtScreenModeRequester *m_screenmoderequester;
    struct rtFileList *m_filelist, *m_tempfilelist;
    struct Hook m_filterhook, m_font_filterhook, m_volume_filterhook;
    APTR m_windowlock;
    ULONG color, longvar;
    char buffer[128], filename[108];
};

////////////////////////////////////////////////////////////////////////////
#endif // __AFREQTOOLS_HPP__
```

1.23 Source

```
////////////////////////////////////////////////////////////////////////////
// ReqTools.cpp - AFrame v1.0 © 1996 Synthetic Input
//
// ReqTools C++ Object utilizing reqtools.library version 38.1296
// by Nico François.  ReqTools is © Nico François
//
//
// Deryk B Robosson
// Jeffry A Worth
// January 20, 1996
////////////////////////////////////////////////////////////////////////////
```

```
  ////////////////////////////////////////////////////////////////////////////
  // INCLUDES
  #include "AFRAME:include/reqtools.hpp"

  #ifndef __AFREQTOOLS_DEFTAGS__
  #define __AFREQTOOLS_DEFTAGS__

  ULONG def_screentags[] = {
RTSC_Flags,SCREQF_DEPTHGAD|SCREQF_SIZEGADS|SCREQF_AUTOSCROLLGAD|SCREQF_OVERSCANGAD ↩
   ,TAG_END } ;
  ULONG def_filetags[] = { RTFI_FilterFunc,TAG_END };
  ULONG def_multifiletags[] = { RTFI_FilterFunc,RTFI_Flags,FREQF_MULTISELECT, ↩
     TAG_END };
  ULONG def_dirtags[] = { RTFI_Flags,FREQF_NOFILES,TAG_END };
  ULONG def_volumetags[] = { RTFI_FilterFunc,RTFI_VolumeRequest,0,TAG_END };
  ULONG def_fonttags[] = { RTFO_FilterFunc,TAG_END };
  ULONG def_palettetags[] = { TAG_END };
  ULONG def_getlong[] = { RTGL_ShowDefault,FALSE,RTGL_Min,0,RTGL_Max,999999, ↩
     TAG_END };
  //ULONG def_getstring[] = {
RTGS_GadFmt,"_Ok|_Cancel",RTGS_BackFill,FALSE,RTGS_Flags,GSREQF_CENTERTEXT| ↩
   GSREQF_HIGHLIGHTTEXT,TAG_MORE,RT_Undersco
re,'_',TAG_END };

  #endif //__AFREQTOOLS_DEFTAGS__

  extern struct ExecBase *SysBase;
  extern struct ReqToolsBase *ReqToolsBase;

  ////////////////////////////////////////////////////////////////////////////
  //

  AFReqTools::AFReqTools()
  {
      color=0;
      filename[0]=0;
      m_filelist=NULL;
      m_screenmoderequester=NULL;
      m_filerequester=NULL;
      m_fontrequester=NULL;
  }

  AFReqTools::~AFReqTools()
  {
      DestroyObject();
  }

  void AFReqTools::DestroyObject()
  {
      if(m_filelist)  rtFreeFileList(m_filelist);
      if(m_screenmoderequester) rtFreeRequest(m_screenmoderequester);
      if(m_filerequester) rtFreeRequest(m_filerequester);
      if(m_fontrequester) rtFreeRequest(m_fontrequester);

      color=0;
      filename[0]=0;
      m_filelist=NULL;
```

```
      m_screenmoderequester=NULL;
      m_filerequester=NULL;
      m_fontrequester=NULL;
  }

  BOOL AFReqTools::RTCreate()
  {
      if(!ReqToolsBase)
          if(!(ReqToolsBase=(struct ReqToolsBase *)OpenLibrary((UBYTE*) ↩
             REQTOOLSNAME,(ULONG)REQTOOLSVERSION)))
              return FALSE;
          else return TRUE;
      else return FALSE;
  }

  BOOL AFReqTools::RTEZRequest(char *text, char *gadfmt)
  {
      return ::rtEZRequest(text,gadfmt,NULL,NULL);
  }

  BOOL AFReqTools::RTEZRequestA(char *text, char *gadfmt, struct rtReqInfo * ↩
     reqinfo, ULONG taglist[])
  {
      return ::rtEZRequest(text,gadfmt,reqinfo,(struct TagItem *)taglist);
  }

  BOOL AFReqTools::RTScreenMode()
  {
      if(SysBase->LibNode.lib_Version<37) {
          rtEZRequestTags("ScreenMode requesters require\n Kickstart 2.0 or
higher.\n","_Ok",NULL,NULL,RT_Underscore,'_',TAG_END);
          return FALSE;
      } else {
          if(m_screenmoderequester=(struct rtScreenModeRequester*)rtAllocRequestA( ↩
             RT_SCREENMODEREQ,NULL)) {
              if(!rtScreenModeRequest(m_screenmoderequester,(char *)"Select a  ↩
                 Screen Mode",(ULONG)RTSC_Flags,
                                      (ULONG)(SCREQF_DEPTHGAD|SCREQF_SIZEGADS| ↩
                                         SCREQF_AUTOSCROLLGAD|SCREQF_OVERSCANGA
D),(ULONG)TAG_END))
                  return FALSE;
              else return TRUE;
          } else {
              rtEZRequest((char *)"Out of memory!",(char *)"Ok!",NULL,NULL);
              return FALSE;
            }
      }
  }

  BOOL AFReqTools::RTScreenModeA(ULONG taglist[])
  {
      if(SysBase->LibNode.lib_Version<37) {
          rtEZRequestTags("ScreenMode requesters require\n Kickstart 2.0 or
higher.\n","_Ok",NULL,NULL,RT_Underscore,'_',TAG_END);
          return FALSE;
      } else {
```

```
          if(m_screenmoderequester=(struct rtScreenModeRequester*)rtAllocRequestA( ↩
             RT_SCREENMODEREQ,NULL)) {
              if(!rtScreenModeRequestA(m_screenmoderequester,(char *)"Select a  ↩
                 Screen Mode",(struct TagItem
*)taglist))
                  return FALSE;
              else return TRUE;
          } else {
              rtEZRequest((char *)"Out of memory!",(char *)"Ok!",NULL,NULL);
              return FALSE;
            }
      }
  }

  BOOL AFReqTools::RTFileRequest()
  {
      if(m_filerequester=(struct rtFileRequester*)rtAllocRequestA(RT_FILEREQ,NULL) ↩
         ) {
          m_filterhook.h_Entry=(ULONG(*)())file_filterfunc;
          if(!rtFileRequest(m_filerequester,filename,"Select a File", ↩
             RTFI_FilterFunc,&m_filterhook,TAG_END))
              return FALSE;
          else return TRUE;
      } else {
          rtEZRequest((char *)"Out of memory!",(char *)"Ok!",NULL,NULL);
          return FALSE;
        }
  }

  BOOL AFReqTools::RTFileRequestA(ULONG taglist[])
  {
      if(m_filerequester=(struct rtFileRequester*)rtAllocRequestA(RT_FILEREQ,NULL) ↩
         ) {
          m_filterhook.h_Entry=(ULONG(*)())file_filterfunc;
          if(!rtFileRequestA(m_filerequester,filename,"Select a File",(struct  ↩
             TagItem *)taglist))
              return FALSE;
          else return TRUE;
      } else {
          rtEZRequest((char *)"Out of memory!",(char *)"Ok!",NULL,NULL);
          return FALSE;
        }
  }

  BOOL AFReqTools::RTFilesRequest()
  {
      if(m_filerequester=(struct rtFileRequester*)rtAllocRequestA(RT_FILEREQ,NULL) ↩
         ) {
          m_filterhook.h_Entry=(ULONG(*)())file_filterfunc;
          m_filelist=(struct rtFileList*)rtFileRequest(m_filerequester,filename," ↩
             Select
Files",RTFI_FilterFunc,&m_filterhook,RTFI_Flags,FREQF_MULTISELECT,TAG_END);
          if(!m_filelist)
              return FALSE;
          else return TRUE;
      } else {
          rtEZRequest((char *)"Out of memory!",(char *)"Ok!",NULL,NULL);
```

```
          return FALSE;
        }
  }

  BOOL AFReqTools::RTFilesRequestA(ULONG taglist[])
  {
      if(m_filerequester=(struct rtFileRequester*)rtAllocRequestA(RT_FILEREQ,NULL) ↩
         ) {
          m_filterhook.h_Entry=(ULONG(*)())file_filterfunc;
          m_filelist=(struct rtFileList*)rtFileRequestA(m_filerequester,filename," ↩
             Select Files",(struct TagItem
*)taglist);
          if(!m_filelist)
              return FALSE;
          else return TRUE;
      } else {
          rtEZRequest((char *)"Out of memory!",(char *)"Ok!",NULL,NULL);
          return FALSE;
        }
  }

  BOOL AFReqTools::RTDirRequest()
  {
      if(m_filerequester=(struct rtFileRequester*)rtAllocRequestA(RT_FILEREQ,NULL) ↩
         ) {
          if(!rtFileRequest(m_filerequester,filename,"Select a Directory", ↩
             RTFI_Flags,FREQF_NOFILES,TAG_END))
              return FALSE;
          else return TRUE;
      } else {
          rtEZRequest((char *)"Out of memory!",(char *)"Ok!",NULL,NULL);
          return FALSE;
        }
  }

  BOOL AFReqTools::RTVolumeRequest()
  {
      if(m_filerequester=(struct rtFileRequester*)rtAllocRequestA(RT_FILEREQ,NULL) ↩
         ) {
          m_volume_filterhook.h_Entry=(ULONG (*)())vol_filterfunc;
          if(!rtFileRequest(m_filerequester,filename,"Select a
Volume",RTFI_FilterFunc,&m_volume_filterhook,RTFI_VolumeRequest,0,TAG_END))
              return FALSE;
          else return TRUE;
      } else {
          rtEZRequest((char *)"Out of memory!",(char *)"Ok!",NULL,NULL);
          return FALSE;
        }
  }

  BOOL AFReqTools::RTFontRequest()
  {
      if(m_fontrequester=(struct rtFontRequester*)rtAllocRequestA(RT_FONTREQ,NULL) ↩
         ) {
          m_fontrequester->Flags=FREQF_STYLE|FREQF_COLORFONTS;
          m_font_filterhook.h_Entry=(ULONG (*)())font_filterfunc;
```

```
        if(!rtFontRequest(m_fontrequester,"Select a Font",RTFO_FilterFunc,& ↩
           m_font_filterhook,TAG_END))
            return FALSE;
        else return TRUE;
    } else {
        rtEZRequest((char *)"Out of memory!",(char *)"Ok!",NULL,NULL);
        return FALSE;
      }
}

BOOL AFReqTools::RTFontRequestA(ULONG taglist[])
{
    if(m_fontrequester=(struct rtFontRequester*)rtAllocRequestA(RT_FONTREQ,NULL) ↩
       ) {
        m_fontrequester->Flags=FREQF_STYLE|FREQF_COLORFONTS;
        m_font_filterhook.h_Entry=(ULONG (*)())font_filterfunc;
        if(!rtFontRequestA(m_fontrequester,"Select a Font",(struct TagItem *) ↩
           taglist))
            return FALSE;
        else return TRUE;
    } else {
        rtEZRequest((char *)"Out of memory!",(char *)"Ok!",NULL,NULL);
        return FALSE;
      }
}

BOOL AFReqTools::RTPaletteRequest()
{
    color=rtPaletteRequest("Change Palette",NULL,TAG_END);

    if(color==-1)
        return FALSE;
    else return TRUE;
}

BOOL AFReqTools::RTPaletteRequestA(ULONG taglist[])
{
    color=rtPaletteRequestA("Change Palette",m_reqinfo,(struct TagItem *)taglist ↩
       );

    if(color==-1)
        return FALSE;
    else return TRUE;
}

void AFReqTools::RTScreenToFront(struct Screen *screen)
{
    rtScreenToFrontSafely(screen);
}

void AFReqTools::RTSetWaitPointer(struct Window *win)
{
    rtSetWaitPointer(win);
}

BOOL AFReqTools::RTLockWindow(struct Window *win)
{
```

```
      m_windowlock=(APTR)rtLockWindow(win);
      return TRUE;
  }

  void AFReqTools::RTUnlockWindow(struct Window *win, APTR winlock)
  {
      rtUnlockWindow(win,winlock);
  }

  ULONG AFReqTools::RTGetLong(char *title, struct rtReqInfo *reqinfo)
  {
      ULONG ret;
      if(!(ret=rtGetLong((ULONG
*)&longvar,title,reqinfo,RTGL_ShowDefault,FALSE,RTGL_Min,0,RTGL_Max,999999,TAG_END ↩
   )))
          return FALSE;
      else return ret;
  }

  ULONG AFReqTools::RTGetLongA(char *title, struct rtReqInfo *reqinfo, ULONG  ↩
     taglist[])
  {
      ULONG ret;
      if(!(ret=rtGetLongA((ULONG *)&longvar,title,reqinfo,(struct TagItem *) ↩
         taglist)))
          return FALSE;
      else return ret;
  }

  ULONG AFReqTools::RTGetString(UBYTE *buffer, ULONG maxchars, char *title, struct ↩
      rtReqInfo *reqinfo)
  {
      ULONG ret;

      if(!(ret=rtGetString(buffer, maxchars, title, reqinfo, RTGS_GadFmt,
"_Ok|_Cancel",RTGS_BackFill,FALSE,RTGS_Flags,GSREQF_CENTERTEXT| ↩
    GSREQF_HIGHLIGHTTEXT,TAG_MORE,RT_Underscore,'_',TAG_E
ND)))
          return FALSE;
      else return ret;
  }

  ULONG AFReqTools::RTGetStringA(UBYTE *buffer, ULONG maxchars, char *title,  ↩
     struct rtReqInfo *reqinfo, ULONG
taglist[])
  {
      ULONG ret;

      if(!(ret=rtGetStringA(buffer, maxchars, title, reqinfo, (struct TagItem *) ↩
         taglist)))
          return FALSE;
      else return ret;
  }

  ULONG AFReqTools::RTGetVScreenSize(struct Screen *screen, ULONG *widthptr, ULONG ↩
      *heightptr)
  {
```

```
    ULONG spacing;

    if(!(spacing=rtGetVScreenSize(screen,widthptr,heightptr)))
        return FALSE;
    else return spacing;
}

BOOL __asm __saveds file_filterfunc (register __a0 struct Hook *filterhook,
       register __a2 struct rtFileRequester *req,
       register __a1 struct FileInfoBlock *fib)
{
    // examine fib to decide if you want this file in the requester
    return TRUE;
}

BOOL __asm __saveds vol_filterfunc (
    REG __a0 struct Hook *hook,
    REG __a2 struct rtFileRequester *filereq,
    REG __a1 struct rtVolumeEntry *volentry
    )
{
    // examine volentry to decide which volumes you want in this requester
    return TRUE;
}

BOOL __asm __saveds font_filterfunc (
    REG __a0 struct Hook *hook,
    REG __a2 struct rtFontRequester *fontreq,
    REG __a1 struct TextAttr *textattr
    )
{
    // examine textattr to decide which fonts you want in this requester
    return TRUE;
}
```

1.24 ReqTools Object Information

```
AFReqTools/Object Information

  The ReqTools Object was written to provide a C++ Object Class interface to the  ↩
     reqtools.library.  Most of the
functions are supported, but alas a few are not because we do not feel they are  ↩
    necessary at this time.  The class
may be updated to support the remaining functions if a need or desire arise.  This ↩
     class is subject to change as
does the reqtools.library.
```

1.25 History

```
                    ***HISTORY***

********************************************************** ReqTools Object v1.0
```

```
Created January 20, 1996 Release January 21, 1996

 - Created all function class objects for reqtools.library
   usage.  The only functions not supported are:

    - rtChangeReqAttrA
    - rtFreeReqBuffer
    - rtReqHandlerA
    - rtSetReqPosition
    - rtSpread

************************************************************
```

1.26 Distribution

```
Distribution

  The programs and files in this distribution are freely distributable, but are  ↩
     also Copyright (c) Jeff Worth and
Deryk Robosson. They may be freely distributed as long as no more than a nominal  ↩
    fee is charged to cover time and
copying costs. AFrame is distributed as non-crippled shareware, it is fully  ↩
    functional.

Commercial Distribution

Commercial usage is allowed if the following conditions are met:

    a) You state in your documentation that your program uses aframe.library
       and that AFrame is Copyright (c) Jeff Worth and Deryk Robosson.

    b) You send us a copy of your finished product(s) using aframe.library.

  If these conditions are met you are allowed to include the Kickstart 2.0 or  ↩
     higher version of aframe.library and
the installation script(s) with your commercial product.



Freely Distibutable Products

  All of the files copyrighted by the authors must remain unmodified.  None of  ↩
     these files may be distributed on its
own, the entire package must be distributed as one whole.  'demo.cpp' is full  ↩
    public domain and can be used in any
way you like.

  There is one exception to the above.  If you plan to release a freely  ↩
     distributable program (either public domain,
freeware or shareware), you may include 'libs/aframe.library', the installation  ↩
    scripts (with icon), the
documentation (with icons) with your distribution.
```

```
If you include AFrame with a crippled shareware program I'd like to ask you to ↩
    send us a full working version.

  Whether your program is freely distributable or commercial, you must state in ↩
     your documentation that your program
uses aframe.library and that AFrame is Copyright (c) Jeff Worth and Deryk Robosson
```

1.27 m_screenmoderequester

```
AFReqTools/m_screenmoderequester

  MEMBER TYPE
   struct rtScreenModeReqester

  DESCRIPTION
   A rtScreenModeRequester struct unique to each class

  SEE ALSO
   RTScreenMode
```

1.28 m_filerequester

```
AFReqTools/m_filerequester

  MEMBER TYPE
   struct rtFileReqester

  DESCRIPTION
   A rtFileRequester struct unique to each class

  SEE ALSO
   RTFileRequest
```

1.29 m_filterhook

```
AFReqTools/m_filterhook

  MEMBER TYPE
   struct Hook

  DESCRIPTION
   A Hook struct used in the rtFileRequester hook

  SEE ALSO
   RTFileRequester
   struct~rtFileRequest
   <utility/hooks.h>
```

1.30 m_volume_filterhook

```
AFReqTools/m_volume_filterhook

  MEMBER TYPE
   struct Hook

  DESCRIPTION
   A Hook struct used in the rtFileRequester hook

  SEE ALSO
   RTVolumeRequest
   struct~rtFileRequest
   <utility/hooks.h>
```

1.31 m_fontrequester

```
AFReqTools/m_fontrequester

  MEMBER TYPE
   struct rtFontReqester

  DESCRIPTION
   A rtFontRequester struct unique to each class

  SEE ALSO
   RTFontRequest
```

1.32 m_font_filterhook

```
AFReqTools/m_font_filterhook

  MEMBER TYPE
   struct Hook

  DESCRIPTION
   A Hook struct used in the rtFontRequester hook

  SEE ALSO
   RTFontRequest
   struct~rtFontRequest
   <utility/hooks.h>
```

1.33 color

```
AFReqTools/color

  MEMBER TYPE
   ULONG
```

```
  DESCRIPTION
   Can be used to retain the color returned by a RTPaletteRequest

  SEE ALSO
   RTPaletteRequest
```

1.34 m_windowlock

```
AFReqTools/m_windowlock

  MEMBER TYPE
   APTR

  DESCRIPTION
   A storage pointer to a locked window to be used with RTUnlockWindow

  SEE ALSO
   RTUnlockWindow
```

1.35 m_reqinfo

```
AFReqTools/m_reqinfo

  MEMBER TYPE
   struct rtReqInfo

  DESCRIPTION
   A rtReqInfo struct unique to each class

  SEE ALSO
   struct~rtReqInfo
```

1.36 buffer

```
AFReqTools/buffer

  MEMBER TYPE
   char

  DESCRIPTION
   Used to retain the string filled by a RTGetString

  SEE ALSO
   RTGetString
```

1.37 m_filelist

```
AFReqTools/m_filelist

  MEMBER TYPE
   struct rtFileList

  DESCRIPTION
   A rtFileList struct unique to each class

  SEE ALSO
   m_tempfilelist
   struct~rtFileList
```

1.38 m_tempfilelist

```
AFReqTools/m_tempfilelist

  MEMBER TYPE
   struct rtFileList

  DESCRIPTION
   A second rtFileList struct which can be used as a buffer

  SEE ALSO
   m_filelist
   struct~rtFileList
```

1.39 filename

```
AFReqTools/filename

  MEMBER TYPE
   char

  DESCRIPTION
   Used to retain the filename filled by a RTFileRequest

  SEE ALSO
   RTFileRequest
   RTVolumeRequest
```

1.40 longvar

```
AFReqTools/longvar

  MEMBER TYPE
   ULONG
```

```
  DESCRIPTION
   Used by RTGetLong and RTGetLongA it is set by the preceeding functions if the  ↩
      user
   enters a number.  If not, then the variable remains NULL.

  SEE ALSO
   RTGetLong, RTGetLongA
   rtGetLong, rtGetLongA
```

1.41 RTEZRequest

```
AFReqTools/RTEZRequest

  NAME RTEZRequest()

    ret = RTEZRequest(char *bodyfmt, char *gadfmt)

    ret = RTEZRequest(char *bodyfmt, char *gadfmt, struct rtReqInfo *reqinfo,
                      ULONG *taglist)

  DESCRIPTION

  INPUTS
    bodyfmt - requester body text, can be format string a la RawDoFmt()
    gadfmt  - text for gadgets (left to right, separated by '|') or NULL
            - *WARNING* click here for information on passing NULL as gadfmt
    reqinfo - pointer to a rtReqInfo structure allocated with rtAllocRequest()
              or NULL

    taglist - if using RTEZRequestA it expects to see a taglist of items for
              customizing the requester.

              EXAMPLE: ULONG taglist[] = { ITEM1, ITEM2, ITEM3, ..., TAG_END };

  NOTE
    See reqtools developer documentation for descriptions of tag items which
    can be used with RTEZRequestA

  RESULT
    ret - 1 (TRUE) for leftmost (positive) response, then each consecutive
          response will return 1 more, the rightmost (false) response will
          return 0 (FALSE), so 1,2,3,...,num-1,0 -- or idcmp flag.

  BUGS
    none known

  SEE ALSO
    struct~rtReqInfo
```