ErrorMsg.readme

COLLABORATORS

	TITLE : ErrorMsg.readme	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 6, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 ErrorMsg.readme **1**

1.1 Informations about errormsg.library . . . 1
1.2 Distribution and disclaimer . . . 1
1.3 Installation of the library . . . 2
1.4 Introduction . . . 3
1.5 ARexx features . . . 3
1.6 The include files . . . 4
1.7 The glue code . . . 4
1.8 ErrorCom and WhatError . . . 5
1.9 Sample OpenLib routine . . . 5
1.10 How to translate errormsg.library . . . 6
1.11 Contact me... . . . 7

Chapter 1

ErrorMsg.readme

1.1 Informations about errormsg.library

```
                Object: errormsg.library
~~~~~~~
Version: 3.03
~~~~~~~~
Date: Mercredi 30 Novembre 1994
~~~~~
Author: Frédéric Delacroix
~~~~~~~
Status: Freeware
~~~~~~~
                             TABLE OF CONTENTS


                Distribution

                Installation

                Introduction
                    Autodocs

                ARexx features

                The include files

                The glue code

                ErrorCom and WhatError

                OpenLib routine

                Translation

                Contact me
```

1.2 Distribution and disclaimer

In this package, you will find version 3.03 of the errormsg.library. I distribute it as a FREEWARE product, meaning that anybody is allowed to copy and spread it as long as the following conditions are met:

- All files remain unchanged. If you have comments to add, do it in a separate file and make sure it is clear I'm not responsible for those. Archiving is of course permitted.

- All files are distributed together. This includes the library file itself, the glue code, include files, the autodoc files, all files in the ErrorCom WhatError and Rexx directories, all icons, the installation script and this file.

- You do not make commercial usage of this library without a written permission from myself. My address can be found at the end of this document.

- If you are the author of a shareware, freeware, giftware, charityware, etc..., you are allowed to distribute the file named errormsg.library along with all the catalogs and the installation script.

- All programs using errormsg.library have a statement in their documentation file, saying that errormsg.library is Copyright 1994 Frédéric Delacroix.

Errormsg.library is freeware, but it remains copyrighted by me. This is not public domain! Collections like Fish's AmigaLib disks, CAM disks are allowed to include errormsg.library in their libraries.

ErrorCom is FreeWare and cannot be spread independently of errormsg.library. WhatError and its source are public domain, and so are the examples, you can do whatever you want with them. The arexx script named "ShowERM.rexx" is public domain, but "ERMID.rexx" is FreeWare.

Of course, I do not make any guarantee of any kind on the correct working of the library or associated programs. You use it entirely at your own risk, as although I did a lot of bug-trapping, I cannot be sure there are no more left.

1.3 Installation of the library

errormsg.library is a runtime library. For it to work correctly, it must be located in the directory that is assigned to LIBS: (usually the Libs directory of your boot disk). As an alternative, you could also use a program like LoadLibrary for a non-definitive installation.

As errormsg.library is localized, you must copy a few catalog files into your LOCALE:Catalogs/<language> directory if you want it to be able to run in a different language than english. You are forced to do this since a library like this has no PROGDIR: assign to find catalogs.

For convenience, I recommend you install the script named "ERMID.rexx" into your REXX: directory. Installation of "ShowERM.rexx" is

optionnal.

All the above is accomplished by the provided installation script. Double-click the icon, and follow the instructions.

1.4 Introduction

All developpers know how boring it is to include in their programs messages telling the user what went wrong on an error. This is a long and tiresome task for the programmer, who would prefer to do more interesting things.
Moreover, including error messages in the executable often dramatically increases the size of the files, and these messages are often in english only. So the idea of a shared library that would provide all localized error messages the system could deliver was born. This is exacly what errormsg.library is. It includes a function to simply get a pointer on the message to display, and functions to display it. As of V2.0, errormsg.library has a query function which enables it to be called from ARexx programs.

In errormsg.library, error messages are identified by a system/subsystem id (to be able to from know who the message is) and the Code itself. Most of the functions provide tags to alter the behaviour of the library. Check the autodoc file for more information.

1.5 ARexx features

As of V2.0, errormsg.library can be an arexx host. For this, you must declare it with the RXLIB command from the Shell:

```
RXLIB errormsg.library 0 -60 2
```

or the AddLib() function whithin ARexx:

```
call addlib('errormsg.library',0,-60,2)
```

As can be noticed above, the query offset for errormsg.library is -60. This is different from the one for amigaguide.library for example so be careful (one mistake and the guru is there !:-). Also please do explicitely request version 2 of the library so that V1 users don't see the guru when ARexx tries to access a non-existing function.

There are currently 5 ARexx functions implemented; they are directly related to the regular functions of the library. See the autodocs for details.

The functions require some arguments, like IDCMP, which are expected in numeric format. To make the writing of ARexx programs easier, I have added a support script. It is named ERMID, and is to be called as a function:

```
Number=ERMID(ID1,ID2,ID3,...) (up to 15 arguments)
```

The arguments taken by this script are keywords that are to be translated into a number, this number can then be used as an argument for errormsg.library (it can be used for something else of course).

The keywords currently implemented are all system and subsystem codes for the library, all memory attributes (not options) (used by the system/subsystem ERMSYS_EXEC/ERMSUB_NoMemory) and IDCMP flags.

If several arguments are given, the resulting values are or'ed into the global result. This will only work for the library flags, mem attributes and IDCMP flags.

To have a clear idea, type this line in the shell, after having added the library to ARexx's environment (as described above), and the script ERMID.rexx to your REXX: directory:

```
rx "say displayerrormsg(ermid(MEMF_CHIP,MEMF_PUBLIC), ermid(ERMSYS_EXEC),
ermid(ERMSUB_NoMemory),'Ha|He|Hi|Ho|Hu','Title',,
ermid(IDCMP_DISKINSERTED,IDCMP_DISKREMOVED))"
```

(do not include the linefeeds, type all on one line). You should then get the message "No enough CHIP memory" in a window with five gadgets, titled "Title", that will disappear when you insert or remove a disk in any drive.

You can then try the demo script, named ShowERM.rexx, that is quite self-explanatory. To specify a code (for error, system or subsystem), you can either select 1,2, or 3 and enter a number, or select 1,2 or 3, enter an identifier string and select 4,5, or 6.

1.6 The include files

I have written the standard include files _lib.i, .i and .h for the library, plus others for the prototypes and pragmas, basing myself on those created by Nico François for his reqtools.library. I have not been able to test them (proto and pragmas), so if you can find any bugs, correct the files and send them to me so I can include the good ones in the next release.

Include files for pascal, oberon, or whatever are also welcome.

1.7 The glue code

The glue code. Well, all I have been able to do is write the source code for the stub routines, as I do not have any tool to make suitable libraries for a C compiler. I suggest someone assembles them into a link library and sends them to me, so I can release them with the next version. Proper credits will of course be given... Thanks!

Note: once again, I took the glue source for reqtools as a reference, thanks to Nico François, author of this wonderful library.

1.8 ErrorCom and WhatError

In the directory ErrorCom, you will find a program named -surprise,surprise- ErrorCom (version 1.07) . It is a commodity that will enable you to see all messages known by errormsg.library, by entering the system and subsystem codes, and the error code in a (nice) gadtools interface. You will also be able to see the effects of DisplayErrorMsgA() and AlertErrorMsg(). Just run the program for a demonstration.

ErrorCom recognizes two keywords, that can be entered either on the command line for the CLI, or as tooltypes from the Workbench (note: CLI requires quotes around the hotkey description, Workbench does not like them). They are CXPRI and SHOWWINKEY, respectively to set the commodities broker priority among the broker list, and to set the hotkey that will be used as show/hide shortcut. The default for the hotkey is "lcommand lshift `", the default priority is 0.

ErrorCom is also a localized program, you will need to copy the required catalog files on your system disk (into the same directory as ErrorCom, or into your LOCALE: one). You can use the provided script. I might write a MUI version of ErrorCom soon.

WhatError is a simple CLI command that enables you to display on the standard output stream (ie the CLI window) an error message with the SYSTEM, CODE and SUBSYSTEM arguments. The source for this command is provided.

WhatError will force the code to 0 if ERMSYS_EXEC/ERMSUB_NoLibrary is used (new for V3.02), ErrorCom will do so only when displaying or alerting.

1.9 Sample OpenLib routine

As of version 3.02, errormsg.library includes a system/subsystem pair named ERMSYS_EXEC/ERMSUB_NoLibrary. It is used to tell the user that a library, device or resource has failed to open. Using a mask of flags (two are currently available: EXECF_NOLIB_USENAME and EXECF_NOLIB_USEVERSION) as the Code argument, you can control the verbosity of the message produced: setting EXECF_NOLIB_USENAME will return a message with a place (formatting code %s) for the name of the library or device, and setting both flags (don't use EXECF_NOLIB_USEVERSION alone) will return a message with a place (formatting code %ld) for the version number of the library. Resources are not opened with a version number, therefore you should not provide a version number in an error message unless you test it by hand (by peeking in the resource structure). Beware that resources do not have to have a version number to work (see their documentation)!

These additionnal arguments can easily be filled with RawDoFmt()-like functions, including EasyRequestArgs() and DisplayMessageA(). Moreover, in order to provide a fully integrated access, the function DisplayErrorMsgA() recognizes two new tags: EMT_LibName (ti_Data points to the library name) and EMT_LibVersion (ti_Data is the version number), to provide these arguments directly.

In order to make it easy to use this powerful feature, I list here (in assembly but easily translatable to C) a routine named OpenLib() whose job is to open the library given in argument and alert the user if something goes wrong.

```
OpenLib ; (D0,Z)Base=OpenLib(LibName,Version)(A1,D0)
  movem.l d3-d4/a6,-(sp)
  move.l  a1,d3
  move.l  d0,d4
  move.l  _ExecBase(pc),a6
  jsr _LVOOpenLibrary(a6)
  tst.l d0
  bne.s .Good
  move.l  #ERMSYS_EXEC,d1
  move.l  #ERMSUB_NoLibrary,d2
  move.l  #EXECF_NOLIB_USENAME!EXECF_NOLIB_USEVERSION,d0
  clr.l -(sp)
  move.l  d3,-(sp)
  pea EMT_LibName
  move.l  d4,-(sp)
  pea EMT_LibVersion
  move.l  sp,a0
  move.l  _ErrorMsgBase(pc),a6
  jsr _LVODisplayErrorMsgA(a6)
  add.l #20,sp
  moveq #0,d0
.Good movem.l (sp)+,d3-d4/a6
  rts
```

Of course, you must not use this routine to open errormsg.library itself ! Instead, if errormsg.library fails to open, you should cause an alert with DisplayAlert(), saying that errormsg.library is required to make your application run. Alternately, if you don't want to open Intuition yourself, you could use exec's Alert() function (with parameters AT_Recovery!AG_OpenLib)...

1.10 How to translate errormsg.library

Errormsg.library and ErrorCom are fully localized, meaning that, provided that locale.library is available, and so are the catalog files, they can be made to run in your language.

However, I do not know any language other than english and french (I forgot almost all my school spanish). So if you want a catalog for your language, you will have to translate the strings yourself.

To do this, just fill in the blanks in the catalog translation files (those files that end with .ct) with the translations of the strings that are in comment. Then send me the resulting file. If all goes well, you will soon receive the compiled catalog. This is valid for either errormsg.library or ErrorCom.

Translations for the doc files or installation scripts are also welcome.

Ooops. Almost forgot: in the translation file for ErrorCom, the gadget labels are expected to begin with the keyboard shortcut letter in upper case (it is not displayed), and the character to be underlined must be preceeded by a "_". For the gadget named "Message", provide "\x01" as the first character (disables the keyboard shortcut).

1.11 Contact me...

I can be contacted for whatever by mail at:

Frédéric Delacroix
5 rue d'Artres
59269 QUERENAING, FRANCE.

I greet all my friends, ProMedia, and AmigaNews.