

ErrorMsg

COLLABORATORS

	<i>TITLE :</i> ErrorMsg		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		December 7, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	ErrorMsg	1
1.1	ErrorMsg.doc	1
1.2	errmsg.library/--rexxhost--	1
1.3	errmsg.library/AlertErrorMsg	2
1.4	errmsg.library/DisplayErrorMsgA	3
1.5	errmsg.library/DisplayMessageA	4
1.6	errmsg.library/GetErrorMsgA	5
1.7	errmsg.library/PutErrorMsg	7

Chapter 1

ErrMsg

1.1 ErrMsg.doc

```
--rexhost--  
AlertErrMsg()  
DisplayErrMsgA()  
DisplayMessageA()  
GetErrMsgA()  
PutErrMsg()
```

1.2 errormsg.library/--rexhost--

HOST INTERFACE

As of V2.0, errormsg.library provides an ARexx function interface that enables ARexx programs to take advantage of the features of ErrMsg. The functions provided by the interface are directly related to the functions described herein, with the differences mostly being in the way they are called.

The function host library vector is located at offset -60 from the library. This is the value you provide to ARexx in the AddLib() function call.

FUNCTIONS

CALL AlertErrMsg(CODE/N/A, SYSTEM/N/A, SUBSYSTEM/N/A)

Num= DisplayErrMsg(CODE/N/A, SYSTEM/N/A, SUBSYSTEM/N/A, GAD, TITLE,
WINDOW/N, IDCMP/N)

Num= DisplayMessage(BODY/A, GAD, TITLE, WINDOW/N, IDCMP/N)

Message= GetErrMsg(CODE/N/A, SYSTEM/N/A, SUBSYSTEM/N/A)

Success= PutErrMsg(CODE/N/A, SYSTEM/N/A, SUBSYSTEM/N/A, HEADER)

NOTES

As these functions are called from ARexx's context, and not from, for example, the CLI that launched the script, the pr_WindowPtr field of the calling process structure may not be representative

of the window the output is actually produced. In fact, it is likely to be 0, in which case the output is produced on the Workbench screen. Since this field provides for the default values of the arguments TITLE and WINDOW, you are likely to get "Information" or "Request" as the title and the window displayed on the default public screen. I might write if you really need it a function that sets the pr_WindowPtr field as needed (by asking the console process), but I'm not sure it is really safe to modify ARexx's context by hand (in fact, I'm sure of the opposite :-).

BUGS

Not really a bug, but it is currently impossible to detect error returns from GetErrMsgA(). May or may not be fixed in the future.

DisplayErrMsg() will force code to 0 when using ERMSYS_EXEC/ERMSUB_NoLibrary because it is currently impossible to pass the library name and version. Might or might not be fixed in the future...

AlertErrMsg() and PutErrMsg() will do so as well, but because of the functions themselves (this is not an ARexx limitation).

1.3 errmsg.library/AlertErrMsg

NAME

AlertErrMsg -- Display a recovery alert with the error message

SYNOPSIS

```
Response=AlertErrMsg(Code, System, SubSystem);
D0          D0    D1    D2
```

```
BOOL AlertErrMsg(LONG, ULONG, ULONG);
```

FUNCTION

This function is your last chance to display an error message to the user: it uses an Intuition recovery alert. This must be really your last chance, as DisplayAlert()ing in a multitasking system is not really polite.

INPUTS

Code - see GetErrMsgA()
 System - see GetErrMsgA()
 SubSystem - see GetErrMsgA()

RESULT

Response - the result of DisplayAlert(). If there is no message for the given System/SubSystem/Code, no alert is produced, and Response is FALSE.

BUGS

See DisplayAlert()
 When using ERMSYS_EXEC/ERMSUB_NoLibrary, the Code argument will be forced to 0 since there's no way of passing in the library name and version as additional args. Use GetErrMsgA() and DisplayAlert() if you really need this feature.

SEE ALSO

GetErrorMsgA(), DisplayAlert()

1.4 errormsg.library/DisplayErrorMsgA

NAME

DisplayErrorMsgA() -- Display an error in a requester
 DisplayErrorMsg() -- varargs stub for DisplayErrorMsgA()

SYNOPSIS

```
Num=DisplayErrorMsgA(Code, System, Subsystem, TagList);
D0          D0   D1   D2   A0

LONG DisplayErrorMsgA(LONG, ULONG, ULONG, struct TagItem *);

Num=DisplayErrorMsg(Code, System, Subsystem, Tag1, ...);

LONG DisplayErrorMsg(LONG, ULONG, ULONG, Tag, ...);
```

FUNCTION

This function is an integrated access to GetErrorMsgA() and DisplayMessageA(). The error message, if it exists, will be displayed in a nice requester. If there is no message for the given system/subsystem/code, no requester is produced, and Num is 0.

As of V3.02, this function handles the special case of library name and version for ERMSYS_EXEC/ERMSUB_NoLibrary.

INPUTS

Code - See GetErrorMsgA()
 System - See GetErrorMsgA()
 Subsystem - See GetErrorMsgA()
 TagList - Pointer to an optionnal tag list, or NULL.

TAGS

All tags understood by either GetErrorMsgA() or DisplayMessageA().
 EMT_LibName (UBYTE *) - The library name for messages from ERMSYS_EXEC/ERMSUB_NoLibrary only.
 Ignored for other systems/subsystems.
 You must set the flag in the code field (see include file for the full description of this procedure).
 EMT_LibVersion (ULONG) - The library version for the same case.
 Set the flag too.

RESULT

Num - See DisplayMessageA(). 0 if there is no message.
 Tags - See GetErrorMsgA() or DisplayErrorMsgA()

BUGS

See EasyRequestArgs()

SEE ALSO

libraries/errmsg.h, GetErrorMsgA(), DisplayMessageA(),

EasyRequestArgs()

1.5 errmsg.library/DisplayMessageA

NAME

DisplayMessageA() -- Display a requester with a message
 DisplayMessage() -- varargs stub for DisplayMessageA()

SYNOPSIS

```
Num=DisplayMessageA(TextFmt,Args,TagList);
D0          A0          A1    A2

LONG DisplayMessageA(UBYTE *,APTR,struct TagItem *);

Num=DisplayMessage(TextFmt,Args,Tag1,...);

LONG DisplayMessage(UBYTE *,APTR,Tag,...);
```

FUNCTION

This function will display an EasyRequest()-style requester with the provided text format, arguments and optionnal taglist.

INPUTS

TextFmt - A text formatting string, exactly as taken by EasyRequestArgs(). Each control character (like %s for a string, %ld for a longword decimal number) will be replaced by the corresponding field in the arguments array.

Args - An array of arguments for the Text format. This can also include arguments for an optionnal gadget format given by the EMT_GadFmt tag. in that case, gadget arguments must follow body text arguments.

TagList - Pointer to an optionnal tag list , or NULL.

TAGS

EMT_Window (struct Window *) - This tag says you want the requester to appear on the same screen as the window identified by ti_Data. It will also set the default title for the requester (see EasyRequest()). The default is to use the window pointed by the pr_WindowPtr of your process structure if it is valid, or the default public screen if it is NULL, or if the calling task is not a process. If pr_WindowPtr is -1, and there is no EMT_Window tag, no requester is produced.

EMT_Title (UBYTE *) - This tag provides a null-terminated string for the title of the requester. The default is the Window title, or "System Request" if none.

EMT_GadFmt (UBYTE *) - This tag says you want to use a custom gadget formatting string. This can include control characters like "%s". The english

default is "Okay.". You can provide several gadget labels by separating them with "|".

You must specify at least one gadget.

EMT_IDCMPPtr (ULONG *) - With this tag, you can make the requester close on receiving one of the IDCMP flags pointed by ti_Data (beware that ti_Data POINTS TO an IDCMP longword).

EMT_MUIBase (APTR) - This tag tells errormsg.library that it should use MUI's requester instead of intuition's. ti_Data must be the base of the muimaster.library (no check is performed).

When using MUI, EMT_IDCMPPtr and EMT_Window are ignored. Moreover, MUI does not support formatting codes in gadget labels (yet ?). New for V3.01.

EMT_MUIAppObject (APTR) - With this tag, you provide a pointer to your application object, as returned by MUI New for V3.01.

EMT_MUIWinObject (APTR) - With this tag, you provide a pointer to the window the requester should appear on. Warning: this is MUI's window object, not Intuition's Window structure !. New for V3.01.

RESULT

Num - This is the number of the gadget clicked. From left to right: 1,2,...,N,0. -1 means that one of the custom IDCMP flags was received. If no requester was produced (pr_WindowPtr equals -1), then Num is 0.

EMT_IDCMPPtr - If Num was -1, the longword pointed by ti_Data contains the IDCMP flag that was received.

NOTE

This is exactly what EasyRequestArgs() returns.

BUGS

See EasyRequestArgs().

SEE ALSO

EasyRequestArgs(), libraries/errmsg.h

1.6 errmsg.library/GetErrorMsgA

NAME

GetErrorMsgA() -- Get a pointer to an error message
GetErrorMsg() -- varargs stub for GetErrorMsgA()

SYNOPSIS

```
message=GetErrorMsgA(Code, System, Subsystem, TagList);
D0          D0    D1    D2    A0
```

```
STRPTR GetErrorMsgA(LONG, ULONG, ULONG, struct TagItem *);
```

```
message=GetErrorMsg(Code, System, Subsystem, Tag1, ...);
```

```
STRPTR GetErrorMsg(LONG, ULONG, ULONG, Tag, ...);
```


FUNCTION

Returns the error message identified by System, Subsystem, Code and modifying tags. If locale.library was not open on initialization, the message will be in english. It will be also the case if the catalog does not exist or cannot be opened.

INPUTS

Code - the error code returned by the system/subsystem. See include files for special cases.
System - a special code identifying the resource that has emitted the error code. Typically, it is a code for a library or a device. They are defined in libraries/errmsg.h
Subsystem - this is system-dependant. Typically, it is the _LVO offset of the function that returned an error for libraries, and the command number for devices. See include file for supported values.
TagList - pointer to an optionnal tag list, or NULL.

TAGS

EMT_Length (ULONG *) - pointer to an ULONG that will be filled with the length of the string.
EMT_Error (ULONG *) - pointer to an ULONG that will hold the error code from errmsg.library if no string can be found. Check include file for defines.
EMT_ForceCopy (APTR) - This tag forces the library to copy the message into the buffer pointed by ti_Data (must be big enough, 128 bytes will do) instead of referencing the internal private message. This lets you build a bigger message or modify it. If this tag is present, message will point at the buffer.

RESULT

message - a pointer to a message string, or NULL if none can be found (this happens for unsupported error codes).
EMT_Length - the ULONG pointed by ti_Data will be filled with the length of the string (or 0 if NULL)
EMT_Error - the ULONG pointed by ti_Data will be filled with an error code if failure, 0 if OK.

NOTE

This function preserves all registers except D0 and D1.
For SetFunction()ners only: D1 points to the name of the system that reported an error. Please do not depend on this value (it is only required for DisplayErrorMsgA()).

BUGS

When you pass ERMSYS_MUI/ERMSUB_MUIAutoError, you must provide the MUIMasterBase as the code parameter. If the base is invalid, you will get the error ERMERR_UnknownCode and a few (harmless) enforcer hits.

SEE ALSO

libraries/errmsg.h

1.7 errormsg.library/PutErrMsg

NAME

PutErrMsg() -- Print an error message

SYNOPSIS

```
Success=PutErrMsg(Code, System, SubSystem, Header);  
D0          D0   D1 D2   A0
```

```
BOOL PrintErrMsg(LONG, ULONG, ULONG, STRPTR);
```

FUNCTION

This function prints the error message identified by the Code, System and SubSystem values to the default output channel using buffered output. It is preceded by the header message and a colon. A linefeed is then added. If there is no corresponding error message, no output is produced, and Success is FALSE. It is also the case if VPrintf() fails.

INPUTS

Code - see GetErrMsgA()
System - see GetErrMsgA()
SubSystem - see GetErrMsgA()
Header - Pointer to an optionnal to header string or NULL for none. Else, the header will be printed before the message itself, and followed by a colon. The string is terminated by a 0.

RESULT

Success - Success/failure code.

NOTE

This function is safe when called from a task, but useless.

BUGS

When using ERMSYS_EXEC/ERMSUB_NoLibrary, the Code argument will be forced to 0 since there's no way of passing in the library name and version as additionnal args. Use GetErrMsgA() and VPrintf() if you really need this feature.

SEE ALSO

GetErrMsgA(), libraries/errormsg.h
