

errmsg

COLLABORATORS

	<i>TITLE :</i> errormsg		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		October 27, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	errmsg	1
1.1	errmsg.doc	1
1.2	errmsg.library/AlertErrorMsg	1
1.3	errmsg.library/DisplayErrorMsgA	2
1.4	errmsg.library/DisplayMessageA	2
1.5	errmsg.library/GetErrorMsgA	4
1.6	errmsg.library/PutErrorMsg	5

Chapter 1

errormsg

1.1 errormsg.doc

```
AlertErrorMsg()  
DisplayErrorMsgA()  
DisplayMessageA()  
GetErrorMsgA()  
PutErrorMsg()
```

1.2 errormsg.library/AlertErrorMsg

NAME

AlertErrorMsg -- Display a recovery alert with the error message

SYNOPSIS

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

```
BOOL AlertErrorMsg(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 GetErrorMsgA()
System - see GetErrorMsgA()
SubSystem - see GetErrorMsgA()

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()

SEE ALSO

GetErrorMsgA(), DisplayAlert()

1.3 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.

INPUTS

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

TAGS

All tags understood by either GetErrorMsgA() or DisplayMessageA().

RESULT

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

BUGS

See EasyRequestArgs()

SEE ALSO

libraries/errormsg.h, GetErrorMsgA(), DisplayMessageA(),
EasyRequestArgs()

1.4 errormsg.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 optional 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 optional gadget format given by the EMT_GadFmt tag. In that case, gadget arguments must follow body text arguments.

TagList - Pointer to an optional 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).

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/errormsg.h

1.5 errormsg.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. There is a special code, 0, that is understood by ERMSYS_DOS/ERMSUB_IoErr (and no other system/subsystem). If you pass this as the Code parameter, then errormsg.library will get the real error code from the IoErr() function (provided that the caller is a process).

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/errormsg.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 `errormsg.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.

BUGS

none known.

SEE ALSO

`libraries/errormsg.h`

1.6 `errormsg.library/PutErrorMsg`

NAME

`PutErrorMsg()` -- Print an error message

SYNOPSIS

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

```
BOOL PrintErrorMsg(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 preceeded 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 `GetErrorMsg()`
`System` - see `GetErrorMsg()`
`SubSystem` - see `GetErrorMsg()`
`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

none known.

SEE ALSO

GetErrorMsgA(), libraries/errormsg.h