Programmers Guide to the E Editor Toolkit. Version 2.0

January 13, 1993

Gennaro (Jerry) Cuomo (GCUOMO at YKTVMV)
Jason Crawford (JASON at YKTVMV)
John Ponzo (JPONZO at YKTVMV)

OS/2 Applications & Tools
IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10958

Contents

Disclaimer - read first!

The sample code for using the current version (5.51) of the E Toolkit and its Enhanced Multi-Line Edit control (E-MLE) is being made available for the benefit of those who need to use it now. It is being released "as-is", with no promise of support, enhancement, or bug fixes. In addition, you should be aware that we are currently working on a new, cleaner, API for the next major update to EPM, which is not necessarily compatible with the one used in this package.

In mentioning the "next major update", we are not guaranteeing that this will ever be released outside of IBM. (That is not a decision that we can make.) We are currently developing EPM 5.60, which includes long line support, pop-up menus, and much more, and EPM 6.00, which is essentially a 32-bit version of EPM 5.60.

If you have a need to develop on top of the newer E-MLE, then please send a note to Gennaro Cuomo (gcuomo at watson.IBM.com) describing why you need it. This will help us demonstrate demand for the new code, and *might* get you on our beta-test list.

Introduction.

The E-Toolkit assists a software developer in building applications that edit multiple lines of text. The E-Toolkit gives developers access to OS/2 functions and PM messages that enable the creation and manipulation of an advanced multi-line edit control window. This multi-line edit window contains an advanced text editor engine based on the "E" text editor technology. (The E Family of text editors include, E3 for DOS, E3AIX for AIX, EOS2 for Full Screen OS/2, EPM and EPM/G for OS/2 Presentation Manager.) This multi-line edit window is referred to as the E-MLE. The E-MLE is a versatile control window that supports many primitive operations which allow easy access to powerful text editing capabilities.

For example, The E-MLE supports the editing of large files (limited only by system memory). Each E-MLE can store multiple files, each of which can be viewed one at a time. (The feature is known as a "file RING") The E-MLE supports text with associated attribute information. Using the attribute feature your application can display text in Multiple Fonts, Multiple colors, and even associate commands with regions of text. (Associating commands with regions of text can be a way of implementing HyperText.) An advanced feature is available which allows other window classes to be overlayed with in a E-MLE. Using this "Overlay Window" feature, a E-MLE can incorporate images along with text.

The E based editor engine runs in its own separate work thread. Therefore, when primitive commands are executed (i.e. Load a file) the main process is not blocked. This built-in feature leads to programs that efficiently use the computers CPU.

The behavior of an E-MLE can be enhanced by adding new methods of behavior in any of the following ways. The E-MLE can be sub-classed using traditional PM functions written in C. (i.e WinSubclassWindow(...)). New methods can also be added to an E-MLE by simply extending event definitions by writing E or REXX macros.

This document describes the Application Program Interface (API) of the E-Toolkit. Examples are provided that demonstrate some of the more popular way in which an E-MLE can be used.

As mentioned above, The E-MLE can also be extended by writing Macros. The E Technical Reference manual describes how macros can be written.

Writting programs using the E Toolkit

There are four basic steps to creating an E-MLE window. The first step is to Register the E-MLE window class. After Registration, a data structure, describing the desired characteristics of your E-MLE, must be created. The next step involves actually creating the E-MLE and an receiving a handle to the E-MLE in return. This handle can now be used to manipulate the E-MLE window.

1) Register the E-MLE window Class.
2) Fill in Data Structure.
3) Create the E-MLE.
4) Manipulate E-MLE

Getting Started.

In order to write application programs using the E-Toolkit you first need to download the latest EPM/G package from OS2TOOLS. You will also need copies of the OS/2 1.3 Programmers ToolKit, and the IBM C/2 compiler. (We currently provide language bindings to the 'C' programming language. ('ETOOLKT.H')

Unpack the FLSBIN files that make up the EPM/G package. (Use LOADRAM to unpack FLSBIN files.)

There are four major pieces that are essential to the building of your own custom editor application.

1) ETKExxx.DLL - Heart of the E-Toolkit
2) ETKRxxx.DLL - Utility Resources
3) E-Macros - Utility Resources
4) Your Application/Sample Programs

ETKExxx.DLL - The "Heart" of the E-Toolkit

The first file is ETKExxx.DLL. This is the dynamic link library that contains the powerful functions that will allow you to create your own custom E-MLE windows. Both messages and functions are available for a user to subclass the behavior of an E-MLE. Sample functions include: EtkInsertText, EtkProcessEditKey, and EtkSetFileField, and EtkQueryText Sample message include the EPM_EDIT_COMMAND message allows any of the editor commands to be issued against the receiving edit window. The API to this library is described in detail in this document.

ETKRxxx.DLL - Utility Resources

The second file of interest is ETKRxxx.DLL. This file contains Utility Resources. We call this our ERES-Toolkit. It contains widgets that enhance productivity when working with E-MLE's. Dialogs boxes such as the Search, Bookmark, and Error Message dialog can be displayed using the EResPopDialog(...) function. Other utility functions like EResCountEditWindow(...) and EResOpenEditWindow(...) are examples of functions provided so programmers need not write some of the most common actions taken upon E-MLE windows. The API to this library is described in detail in this document.

E-Macros - Utility Resources

The set of E macros files can be use to build your own custom macros which will enhance your applications E-MLE. For example, you can build new commands using the E macros "defc" construct that can be executed in C using either the EtkExecuteCommand function of the EPM_EDIT_COMMAND message.

Your Application/Sample Programs

This is the program that you write. For example EPM.EXE. EPM.EXE is small piece of code that creates E-MLE's and provides the proper glue needed to tie in dialogs, help, macros, and etc. (The *source code* to EPM.EXE ver 5.50 is available via request to EOS2. It will become a permanent piece of our toolkit package.)

Note: It is important that the two dynalink libraries are placed in a sub-directory that is specified in your LIBPATH environment variable. (Your LIBPATH environment variable is found in your CONFIG.SYS file.)

Before you attempt to write you own application, you should examine the sample programs provided with this package. A sample program named ESIMPLE is provided to demonstrate the "basics" of an E-MLE application. The sample contains two source files:

- ESIMPLE.C
- ESIMPLE.E

ESIMPLE.C creates a parent window containing a single E-MLE window. The parent monitors cursor movement within the E-MLE by processing the EPM_EDIT_CURSORMOVE message. When this message is received, a status line is painted containing the line and column of the cursor.

ESIMPLE.E contains a definition of a handful of keys.

ESIMPLE.MAK is provided to compile and link all components of the ESIMPLE example. (When you build your own custom application, be sure to use the same link and compile options as used in the example.

You are now ready to write your own editor application using the Editor Toolkit.

Special Notes!!!

1) PM Queue size.

   Take note of the PM Queue Size specified in the ESIMPLE program.

   ```
   hmq = WinCreateMsgQueue(hab,  x1  );   // Create Queue
   ```

   We strongly recommend the above PM queue definition because the E-MLE requires a large queue for message passing.

2) E-MLE window handles. (E-MLE Positioning)

   The E-MLE window is a standard PM window. At minimum, it contains a frame window as well as a client window. (You can us the PMstyle flag to add any other standard window control as well.) To position the E-MLE window one must use the frame window handle. The EtkCreate function returns a handle to the E-MLE client window. The following is an example of positioning an E-MLE window.

   ```
   VOID PositionMLE( HWND hwndMLE, SHORT x, SHORT y, SHORT cx, SHORT cy)
   { HWND hMLEFrame=WinQueryWindow( hwndMLE, QW_PARENT, FALSE );
     WinSetWindowPos(hMLEFrame, NULL, x ,y ,cx ,cy, SWP_MOVE | SWP_SIZE)
   }
   ```

E Toolkit functions at a glance.

The following functions are used to create and manage E-MultiLine Edit (E-MLE) windows:

Register, Create, Delete functions (The basics)

The following Dynalink functions are used to registration, initialization, and destroy E-MLE windows.

EtkRegister
: Register the E-MLE window class.

EtkCreate
: Create an E-MLE Window.

EtkDestroy
: Destroy a Edit Window.

EtkVersion
: Retrieve the ETKxxx.DLL version number.

EtkGetProcAddrs
: Retrieve function address of the "basic" functions.

Special Access- E-MLE Manipulation

The following Dynalink functions are used to manipulate various aspects of an E-MLE.

Event Execution.

EtkExecuteCommand
: Execute an command

EtkProcessEditKey
: Execute a "built-in" key action

Text Minipulation

EtkDeleteText
: Delete a line of text.

EtkReplaceText
: Replace a line of text with a new line.

EtkInsertText
: Insert a new line of text.

EtkQueryText
: Retreive a line of text.

EtkQueryTextBuffer
: Retreive a range of lines.

EtkInsertTextBuffer
: Insert a stream of text.

EtkFindAttribute
: Search for an attribute associated with a line.

Text Selection

EtkSetSelection
Select text
EtkQuerySelection
Query selection region.
EtkQuerySelectionType
Query Selection Type.

File Information

EtkQueryFileID
Query the active files id.
EtkSetFileField
Set file related information
EtkQueryFileField
Query file related information
EtkQueryFileFieldString
Query file related information

Fonts

EtkRegisterFont
Register a new font
EtkRegisterFont2
Register a new fixed pitch font.

Special Access.

EtkAccessLowLevelData
It's a secret. :-)

Register, Create, Delete functions (The basics)

EtkRegister - (Register an E-Multi-line Edit Window)

Register the E-Multi-Line Edit Window Class for this process.

EtkRegister (hab, class_style, *Window_Class_Name* **)**

Parameters

hab *(HAB)* - input
Application anchor block

class_style *(ULONG)* - input
Default window style. This can be any of the standard PM class styles CS_*.

Returns

Window_Class_Name *(PSZ)* -
Pointer to window-class-name string in which the editor window was registered under.

Remarks

This call only needs to be called once per application. This function issues a WinRegisterClass. Your application can customize this registration process. The first 16 bytes of window storage is reserved for the E-MLE window. You can use EtkWndProc as the default window procedure for your custom class.

Example

```
WinRegisterClass(                       // Register Window Class
                 (HAB)hab,              // Anchor block handle
                 (PSZ)classEwindow,       // Window Class name
                 (PFNWP)EtkWndProc,       // Address of Window Procedure
                 (ULONG)class_style,      // class style
                 (USHORT)16             // ETK uses the first 16bytes of
                );                      // the reserved long.
```

EtkCreate - (Create a E-Multi-Line Edit Control Window)

Create a new E-Multi-line edit control window containing specified file(s).

EtkCreate (epm_p, hwnd_p, *rc* **)**

Parameters

epm_p *(PEDITORINFO)* - input

Pointer to Text Editor Information Structure.

```
.-------------------------------------------------------------------
|  Editor Information Structure defined in 'C'
|-------------------------------------------------------------------
| typedef struct EDIT_INFO_TYPE {
|   HAB      hab;          // application anchor block
|   HWND     hwndparent;    // handle to parent of edit window
|   HWND     hwndowner;     // handle to owner of edit window
|   PRECTL    pswp          // positioning of edit window
|   PSZ      filename;      // file to be edited (with wildcard)
|   HPOINTER  hEditPtr;      // handle to editor pointer icon.
|   HPOINTER  hMarkPtr;      // handle to mark pointer icon.
|   PVOID    hEditorIcon;   // editor ICON.
|   ULONG    editorstyle;   // internal editor options
|   ULONG    pmstyle;       // PM standard window styles (FCF_xxxx)
|   USHORT    font;         // - Not used in versions greater than 5.2
|   PSZ      exfile;       // pre-compiled macro code file (EPM.EX)
|   PSZ      *topmkr;       // top and bottom of file marker
|   PSZ      *botmkr;       //                                      |
|   SHORT    editid;        // unique window id specified for edit w
|   PSZ      exsearchpath;  // a set of paths to search for ex's fi
|   PSZ      exe_path;      // path where the application started
|   HINI     hini;         // handle to an opened ini profile
| } EDITORINFO;                                                     |
|                                                                   |
'-------------------------------------------------------------------
```

See Appendix A for a detailed description of the EDITORINFO structure.

hwnd_p *(PHWND)* - output

Pointer to E-MLE's client window.

Returns

rc *(USHORT)* -

TRUE if a error occurred during E-MLE window creation.

Remarks

This call creates a unique E-MLE window. Each E-MLE is a complete new instance of the text editor. Therefore, it knows little about any other E-MLE windows that have been previously created. Each E-MLE window has it's own marks, tabs, margins, and etc.

Starting with version 5.50, the E-MLE's use GPI fonts for text I/O. An E-MLE window can render text in multiple font faces and sizes. (See Appendix ???????)

An application can create as many E-MLE's as system memory will permit. Each E-MLE instance can contain a ring a files. A RING can contain as many files as system memory will permit.

EtkDestroy - (Destroy a Multi-line Edit Control Window)

Close a E-MLE window.

EtkDestroy (hab, hwndOwner, hwndEdit, *rc* **)**

Parameters

hab *(HAB)* - input
Anchor block

hwndOwner *(HWND)* - input
Handle of the owner of the E-MLE window. Usually this is the application client window.

hwndEdit *(HWND)* - input
E-MLE window to destroy

Returns

rc *(USHORT)* -
One of the following messages are returned. (These messages are defined in EDLL.H)

- EPM_RC_DESTROYOK - E-MLE window has successfully closed.
- EPM_RC_DESTROYTIMEOUT - the interpreter has timed out because it was either running a very long command, or it was executing a defexit that had a problem.
- EPM_RC_DESTROYCANCEL - the user did not wish to close the E-MLE window.

Remarks

Since E is a memory based editor, the destroy function will not effect the most recent copy of the text file on disk. The destroy function generates a WM_DESTROY message. It is this message that actually closes the edit window. You may choose to send this message directly instead of using the destroy function. It is important to check the return code when sending the E-MLE window a WM_DESTROY. If the E-MLE window receives a destroy message, and a modified file exists in its ring, an EPM_EDIT_ASKTOCLOSE message is sent to the application. (See EPM_EDIT_ASKTOCLOSE message in the section "Editor Toolkit Messages") This message gives the application a chance to display a dialog box asking the user if they want to "really close" the window.

EtkVersion - (Dynalink Version)

Get the version of the Editor dynalink library(ETKExxx.DLL)

EtkVersion (version_str,)

Parameters

version_str *(PSZ)* - output

Pointer to a buffer of memory. This will be filled with the current dynalink version. The return string is in the following format:

```
byte [1]         - length of string to follow.
byte [2 - length]  - Null terminated version string.
```

Returns

VOID

Remarks

A constant, 'EVERSION', is provided in the EDLL.H file. This constant is to be compared to the results of the EtkVersion procedure. An example follows:

Example

```
CHAR EDLLVersion[2 ];

EtkVersion( EDLLVersion );            // Get Version from .DLL
vercmp=strcmp( EDLLVersion, EVERSION ); // compare version numbers
if (vercmp) {
   WinMessageBox ((HWND)HWND_DESKTOP,   // Display editor return cod
                  (HWND)hwndAppFrame,
                  (PSZ)"Version Mismatch",
                  (PSZ)"Check ETKExxx.DLL version",
                  NULL,
                  MB_ICONEXCLAMATION
                 );
   return(1);
}
```

Special Access Functions- E-MLE Manipulation

Event Execution.

EtkExecuteCommand - (Execute an E -Command)

This function will execute any predefined command.

EtkExecuteCommand (hwndEdit, command, *rc* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle to the E-MLE in which the specified command will execute.

command *(PSZ)* - input
An asciiz command string that is defined as either an internal command, an E macro, or a REXX macro.

Returns

rc *(ULONG)* -
TRUE if action failed.

Remarks

This is one of the more versatile commands available in the E Toolkit. Three classes of commands can be executed:

- Internal Commands. (examples - SAVE, L[ocate], and C hange“).
- Predefined E macros. (examples - ADD, DRAW, and GET)
- REXX macros commands. (examples - RX SORT.ERX, RX ADDMENU.ERX, RX NEW.ERX)

An example of how to use this function follows.

Example

```
{
 HWND hwndEdit=QueryEditHandle( );

 // issue a "save" command to an E-MLE.  The data is saved in
 // a file called temp.tmp.
 EtkExecuteCommand( hwndEdit, "SAVE temp.tmp") ;
}
```

EtkProcessEditKey - (Execute a editor action)

Execute a primitive editor action.

EtkProcessEditKey (hwndEdit, key, *rc* **)**

hwndEdit *(HWND)* - input
Window handle of the E-MLE.

key *(USHORT)* - input
Built in key primitives:

ADJUST_BLOCK
BACKTAB Tab to the left.
BACKTAB_WORD Tab to the left one word.
BEGIN_LINE Move the cursor to the beginning of the line.
BOTTOM Move the cursor to the last line in the file.
COPY_MARK Copy the current selection to the cursor location.
DELETE_CHAR Delete the character in which the cursor is located.
DELETE_LINE Delete the line in which the cursor is located.
DELETE_MARK Delete the current selection.
DOWN Scroll down one line.
END_LINE Move the cursor to the end of the current line.
ERASE_END_LINE Erase from the cursor to the end of the line.
INSERT_LINE Insert a new line after the cursor.
INSERT_TOGGLE Toggle between insert and replace cursor.
JOIN Join next line with the line in which the cursor is located.
LEFT Scroll one character to the left.
MOVE_MARK Move the current selection to the cursor location.
NEXT_FILE Go to the next file in the Ring.
OVERLAY_BLOCK Overlay the current block selection to the cursor location.
PAGE_DOWN Scroll one page down.
PAGE_UP Scroll one page up.
PREVFILE_OP Go to the previous file in the Ring.
REFLOW Reflow the text according to the margins settings.
REPEAT_FIND Repeat the last find.
RIGHT Scroll right one character.
RUBOUT Move cursor to the left and delete character.
SHIFT_LEFT Shift the selected text one character to the left.
SHIFT_RIGHT Shift the selected text one character to the right.
SPLIT Split the current line at the cursor.
TAB Move cursor to the next tab position.
TAB_WORD Tab to the right one word.
TOP Move the cursor to the first line in the file.
UNDO_LINE Return the current line to its original state.
UNMARK Clear the current selection.
UP Scroll up one line.

Remarks

Example

Text Manipulation

EtkQueryText - (Retrieve a line)

Retrieve a line of text and its associated attributes.

EtkQueryText (hwndEdit, fileid, linenum, Text, Attrs, ALAttr, *rc* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle to the E-MLE that will contain the selected text.

fileid *(ULONG)* - input
the fileid of the file containing the specified line. If 0 is specified, the line will be retrieved from the visible file.

linenum *(ULONG)* - input
line number of interest.

Text *(PSZ *)* - output
Returns an indirect pointer to the text found on the specified line.

Attrs *(PVOID)* - output
returns a pointer to an attribute structure that describes the attributes (i.e. font, color) associated with that line.

ALAttrs *(PVOID)* - output
returns a pointer to an attribute structure that describes the attributes (i.e. font, color) associated with that line.

Returns

rc *(ULONG)* -
TRUE if action failed.

Remarks

```
*****************************************************************
***  TODO -  Need a section on how to interpret the attribute struct
*****************************************************************
```

Example

EtkDeleteText - (Delete text)

Delete an area of text.

(hwndEdit, fileid, startline, number_oflines, *rc* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle to the E-MLE.

fileid *(ULONG)* - input
the fileid of the file containing the specified line. If filid=0 is specified, the line will be retrieved from the visible file.

startline *(ULONG)* - input
starting line number

num_of_lines *(ULONG)* - input
number of lines to delete. If (-1L) then lines are deleted from the start line through the last line in the file.

Returns

rc *(ULONG)* -
TRUE if action failed.

Remarks

Example

EtkReplaceText - (Replace a line)

Replace a line with a new single attributed string

EtkReplaceText (hwndEdit, fileid, linenum, AttrString, *rc* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle to the E-MLE that will contain the selected text.

fileid *(ULONG)* - input
the fileid of the file containing the specified line. If 0 is specified, the line will be retrieved from the visible file.

linenum *(ULONG)* - input
starting line number

Attrs *(PVOID)* - input
a pointer to an attribute structure that describes the attributes (i.e. font, color) associated with that line.

Returns

rc *(ULONG)* -
TRUE if action failed.

Remarks

```
*****************************************************************
***  TODO -  Need a section on how to interpret the attribute struct
*****************************************************************
```

Example

EtkInsertText - (Add a line)

Insert a single attributed string

EtkInsertText (hwndEdit, fileid, linenum, AttrString, *rc* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle to the E-MLE that will contain the selected text.

fileid *(ULONG)* - input
the fileid of the file containing the specified line. If 0 is specified, the line will be retrieved from the visible file.

linenum *(ULONG)* - input
starting line number

Attrs *(PVOID)* - input
A pointer to an attribute structure that describes the attributes. (i.e. font, color) associated with that line.

Returns

rc *(ULONG)* -
TRUE if action failed.

Remarks

```
*********************************************************************
***  TODO -  Need a section on how to interpret the attribute structur
*********************************************************************
```

Example

EtkQueryTextBuffer - (Query Text Buffer)

Retrieve a stream of text within a specified line range.

EtkQueryTextBuffer (hwndEdit, startline, lastline, TotalLen, buffer, *characters* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle of the E-MLE.

startline *(ULONG)* - input
First line in text range.

lastline *(ULONG)* - input
Last Line in range.

TotalLen *(ULONG)* - input
Size of buffer

buffer *(PSZ)* - output
Buffer where text stream will be returned.

Returns

characters *(ULONG)* -
The number of characters in the buffer is returned. If 0 is returned, it is likely that a problem occured. (Most likely an invalid line number)

Remarks

If startline and lastline are zero (0), the entire file will be returned.

The Buffer contains a stream of characters where each line ends in CR-LF.

EtkInsertTextBuffer - (Insert Text Buffer)

Enter a stream of text starting at a specified point.

EtkInsertTextBuffer (hwndEdit, startline, TotalLen, Text, *lines* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle of the E-MLE.

startline *(ULONG)* - input
Insert starting at this line.

TotalLen *(ULONG)* - input
Size of buffer

Text *(PSZ)* - input
Text stream to insert.

Returns

lines *(ULONG)* -
The number of lines inserted in to the E-MLE. If 0 is returned, it is likely that a problem occured. (Most likely an invalid line number)

Remarks

If startline is zero (0), the Text is inserted at the current cursor position.

The Text buffer should contain a stream of characters where each line ends in CR-LF.

Example

```
{
 UCHAR string[512],
 strcpy(string, "1: This is a test\r\n2: This is line 2\r\n");
 EtkInsertTextBuffer( hwndEdit, 1, strlen(string), string);
}
```

EtkFindAttribute. - (Retrieve a line attribute)

Search for an attribute associated with a line.

EtkFindAttribute (hwndEdit, fileid, linenum, Column, ColumnOffset, Attribute, pFound, *rc* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle to the E-MLE that will contain the selected text.

fileid *(ULONG)* - input
the fileid of the file containing the specified line. If 0 is specified, the line will be retrieved from the visible file.

linenum *(ULONG)* - input
starting line number

Column *(USHORT)* - input
starting column

ColumnOffset *(USHORT)* - input
Column Offset.

Attrs *(PVOID)* - input
A pointer to an attribute structure that describes the attributes (i.e. font, color) associated with that line.

pFound *(BOOLEAN *)* - output
found flag.

Returns

rc *(ULONG)* -
TRUE if action failed.

Remarks

```
*********************************************************************
***  TODO -  Need a section on how to interpret the attribute struct
*********************************************************************
```

Example

Text Selection.

EtkQuerySelectionType - (Query Current Selection Type)

Returns the current selection type. The E Toolkit supports three types of text selection. Character, Line, and Block.

EtkQuerySelectionType (hwndEdit, marktype, *rc* **)**

Parameters

hwndEdit *(HWND)* - input

Window handle of the E-MLE.

marktype *(PUSHORT)* - output

The E Toolkit will fill this value with a constant that represents the type of mark contained in the visible file.

```
#define LINEMARK
#define CHARMARK    1
#define BLOCKMARK   2

#define CHARMARKG   3
#define BLOCKMARKG  4
#define NOMARK     -1
```

Returns

rc *(USHORT)* -

TRUE if action failed.

Remarks

Example

```
{
  HWND   hwndEdit=QueryEditHandle( );
  USHORT  marktype;
  UCHAR   outmsg[MAXSTR];

  strcpy( outmsg, "Your file contains ");
  EtkQuerySelectionType( hwndEdit, &marktype);
  switch (marktype) {
     case  LINEMARK:   strcat( outmsg, "a Line Mark");           br
     case  CHARMARK:   strcat( outmsg, "a Character Mark");       b
     case  CHARMARKG:  strcat( outmsg, "a Special Character Mark");
     case  BLOCKMARK:  strcat( outmsg, "a Block Mark");          br
     case  BLOCKMARKG: strcat( outmsg, "a Special Block Mark");
     default:
          strcat( outmsg, "no mark.");
          break;
  }

}
```

EtkQuerySelection - (Selection Location)

Returns the coordinates of the selected text area.

EtkQuerySelection (hwndEdit, firstline, lastline, firstcol, lastcol, markfileid, respectattributes, relative2file, *rc* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle of the E-MLE that contains selected text.

firstline, lastline *(PULONG)* - output
returns the first and last lines in the selected area.

firstcol, lastcol *(PUSHORT)* - output
returns the first and last columns in the selected area.

markfileid *(PULONG)* - output
returns the fileid of the file containing the selected text.

respectattributes *(PUSHORT)* - output
If TRUE, Font and color attribute information will be taken into consideration.

relative2file *(PUSHORT)* - output
Should always be set to TRUE.

Returns

rc *(USHORT)* -
TRUE if action failed.

Remarks

Example

```
{
 ULONG  firstline, lastline;
 USHORT firstcol, lastcol;
 LONG   markfileid;
 SHORT  marktype;
 USHORT respectattributes=FALSE, relative2file=TRUE;

 EtkQuerySelectionType( hwndEdit, &marktype);

 if (marktype!=NOMARK) {
    EtkQuerySelection(QueryActiveEdit(),
                  &firstline, &lastline,
                  &firstcol,  &lastcol,
                  &markfileid,
                  respectattributes, relative2file);
 }
}
```

EtkSetSelection - (Select Text)

Specifiy the type of selection along with the area of text to be selected.

EtkSetSelection (hwndEdit, firstline, lastline, firstcol, lastcol, firstoff, lastoff, marktype, fileid, *rc* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle to the E-MLE that will contain the selected text.

firstline, lastline *(ULONG)* - input
The first and last lines in the selected area.

firstcol, lastcol *(USHORT)* - input
The first and last columns in the selected area.

firstoff, lastoff *(USHORT)* - input
The attribute offset associated with the first and last column. (If color or font attributes are not being used, set to 0)

marktype *(USHORT)* - input

```
#define LINEMARK
#define CHARMARK    1
#define BLOCKMARK   2

#define CHARMARKG   3
#define BLOCKMARKG  4
#define NOMARK     -1
```

markfileid *(USHORT)* - input
the fileid of the file containing the selected text. If -1 is specified, the text will be selected in the visible file.

Returns

rc *(USHORT)* -
TRUE if action failed.

Remarks

Example

```
{
 ULONG  firstline, lastline;
 USHORT firstcol, lastcol;
 LONG   markfileid;
 SHORT  marktype;
 USHORT respectattributes=FALSE, relative2file=TRUE;

 EtkQuerySelectionType( hwndEdit, &marktype);

 if (marktype!=NOMARK) {
    HWND hwndEdit=QueryActiveEdit();
    EtkQuerySelection(hwndEdit,
                  &firstline, &lastline,
                  &firstcol,  &lastcol,
                  &markfileid,
                  respectattributes, relative2file);
    // extend the mark one line.
    EtkSetSelection(hwndEdit,
                  firstline, lastline+1,
                  firstcol,  lastcol,   ,  ,
                  marktype,
                  markfileid);
 }
}
```

File Information

EtkQueryFileID - (Query a files identifier)

Retrieve the visible files file identifier.

EtkQueryFileID (hwndEdit, fileid, *rc* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle to the E-MLE that will contain the selected text.

fileid *(PUSHORT)* - input
returns the fileid of the visible file.

Returns

rc *(USHORT)* -
TRUE if action failed.

Remarks

The file identifier is an constant that is used by certian functions to identify a particular file in an E-MLE window Ring.

Example

EtkSetFileField - Set some file characteristic

Set information pertaining to some aspect of a file. (i.e. Margin, Name, etc)

EtkSetFileField (hwndEdit, field, fileid, indata, *rc* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle of the E-MLE.

field. *(ULONG)* - input
See Appendix A1 for field constants and values expected by each field.

fileid *(ULONG)* - input
the fileid

indata *(PVOID)* - input
The input data varies depending on the field being set. **See Appendix A-1 for field constants and values expected by each field.**

Returns

rc *(USHORT)* -
TRUE if action failed.

Remarks

Example

```
{
  UCHAR s[MAXSTR];
  ULONG Browse;

  Browse =  L;   // turn OFF browse mode - the file can now be mod
  EtkSetFileField(hwndMLE,(ULONG)READONLY_FIELD,(ULONG) ,(PVOID)B

  // Insert a stream of text...
  strcpy( s,"Hello World\r\nAnother Line\r\nAnd another...");
  EtkInsertTextBuffer( hwndMLE, 1, strlen(s), s);

  Browse = 1L;   // turn ON browse mode - the file is now read onl
  EtkSetFileField(hwndMLE,(ULONG)READONLY_FIELD,(ULONG) ,(PVOID)B
}
```

EtkQueryFileField - (Query some file characteristic)

Query information pertaining to some aspect of a file. (i.e. Margin, Name, etc)

EtkQueryFileField (hwndEdit, field, fileid, returndata, *rc* **)**

Parameters

hwndEdit *(HWND)* - input
Window handle of the E-MLE.

field. *(ULONG)* - input
See Appendix A1 for field constants and values expected by each field.

fileid *(ULONG)* - input
the fileid

returndata *(PVOID)* - output
data returned. This data varies depending on the field being queried. **See Appendix A1 for field constants and values expected by each field.**

Returns

rc *(USHORT)* -
TRUE if action failed.

Remarks

Example

```
{
  ULONG numlines;

  // query the number of lines in a given file.
  EtkQueryFileField(hwndMLE,(ULONG)LAST_FIELD,(ULONG) ,(PVOID)&numli
}
```

E Toolkit Messages

Purpose The following messages are used to interact with an edit window object.

Remarks

The numbering of the following messages are: (using C and PM standards)

```
EPM_BROADCASTHELP
EPM_EDIT_OPTION
EPM_EDIT_PLAYKEY
EPM_EDIT_ACTIVATEFILEID
EPM_EDIT_POSTDONE
EPM_EDIT_ACTIVEHWND
EPM_EDIT_ASKTOCLOSE
EPM_EDIT_ASKTODONE
EPM_EDIT_ASKTOFAILED
EPM_EDIT_QUERYRECORDKEY
EPM_EDIT_ASKTOQUIT
EPM_EDIT_QUERY_HELP_INSTANCE
EPM_EDIT_CHANGEFONT * (to be implemented for 5.5 )
EPM_EDIT_RECORDKEY
EPM_EDIT_CHAR
EPM_EDIT_RETCODE
EPM_EDIT_CLIPBOARDCOPY
EPM_EDIT_SAYERROR
EPM_EDIT_CLIPBOARDPASTE
EPM_EDIT_COMMAND
EPM_EDIT_SHOW
EPM_EDIT_COMMAND2
EPM_EDIT_TURN_OFF_HIGHLIGHT
EPM_EDIT_CONTROLTOGGLE * ( need access to CONTROL constants.)
EPM_EDIT_CURSORMOVE
EPM_EDIT_VERSION
EPM_EDIT_WIN2DOC * (Jason needs to document this one.)
EPM_EDIT_DESTROYNOTIFY
EPM_EDIT_DESTROYRC
EPM_EXTRAWINDOW_REFRESH
EPM_EDIT_DOC2WIN * (Jason needs to document this one.)
EPM_FRAME_MESSAGELINE
EPM_FRAME_STATUSLINE
EPM_EDIT_ENDRECORDKEY
EPM_EDIT_TASKLIST

------------- TODO ----------------
EPM_EDIT_EXEC_DYNALINK * (GAC has to document this one.)
EPM_GET_ERROR_MESSAGE
EPM_EDIT_EXEC_PROC
EPM_IS_HELP_LOADED
EPM_EDIT_GETMEM
EPM_PRINT_RENDERPAGE
EPM_EDIT_GETPROFILE
EPM_PRINT_RENDERPAGERC
EPM_EDIT_HELPNOTIFY
EPM_QHELP_TABLE
EPM_EDIT_ID
EPM_QUERY_GLOBDATA
EPM_EDIT_MINMAXFRAME
EPM_SEND_MACROS_ERRORS
EPM_EDIT_NEWFILE
-----------------------------------
```

Note: Editor messages are either ***send*** messages or ***receive*** messages. The ***send*** messages are sent from your application to the edit window. The ***receive*** messages are sent by an edit window to the registered owner's window procedure.

EPM_FRAME_STATUSLINE

(Update Status Line) - Sent TO an E-MLE window

Purpose Updates the status line's template and color.

Message Parameter 1

status_template (PSZ) Status temple string to be filled in by status line. (See Remarks for status_template format)

Message Parameter 2

statusline_color(USHORT)

(LOCHAR)foreground color
(HICHAR)background color

Return

NULL

Remarks: If a color is not specified, then the current color is used. *The pointer passed in Message Parameter 1 IS FREED by this message.* It must be allocated using DosAllocSeg, because it is freed using DosFreeSeg.

The following tags are supported in the status template:

%l	current line number.
%c	current column number.
%f	files in E-MLE Ring.
%m	displays "Modified" if visible file is modified.
%i	Displays "Insert" or "Replace"
%z - %x	Displays character at cursor is ascii(%z) or in hex(%x).
%s	Total number of lines in file.
%a	Autosave value.

For example, if the following template string was passed:

```
Line=%l  Column=%C  Mode=%i,  The Ring contains %f
```

The status line would display:

```
Line=129   Column=72  Mode=Insert, The Ring contains 2 Files.
```

EPM_FRAME_MESSAGELINE

(Update Message Line) - Sent TO an E-MLE window

Purpose Updates the message line's text and color.

Message Parameter 1

message (PSZ) string to appear on message line

Message Parameter 2

messageline color(USHORT)

(LOCHAR)foreground color
(HICHAR)background color

Return

NULL

Remarks: If a color is not specified, then the current color is used. *The pointer passed in Message Parameter 1 IS FREED by this message.* It must be allocated using DosAllocSeg, because it is freed using DosFreeSeg.

EPM_EDIT_VERSION

(E-MLE Version) - Sent TO an E-MLE window

Purpose Return the E-Toolkit version number.

Message Parameter 1

not used (LONG)

Message Parameter 2

not used()

Return

NULL

Remarks:

EPM_EDIT_TURN_OFF_HIGHLIGHT

(Turn Off Highlight) - Sent TO an E-MLE window

Purpose Clear text circled by the CIRCLEIT macro.

Message Parameter 1

not used (LONG)

Message Parameter 2

not used()

Return

NULL

Remarks:

EPM_EDIT_QUERY_HELP_INSTANCE

(Query Help Instance) - Sent FROM an E-MLE window

Purpose The following is a request from the etoolkit for the handle of the help instance

Message Parameter 1

not used (LONG)

Message Parameter 2

not used()

Return

NULL

Remarks: This is how EPM.EXE handles this messages.

```
case EPM_EDIT_QUERY_HELP_INSTANCE:
  if(!GlobData->hwndHelpInstance) {
    GlobData->hwndHelpInstance = WinCreateHelpInstance(GlobData->hAB,&GlobData->hmiHelpData);
  }
  return(GlobData->hwndHelpInstance);
  break;
```

EPM_BROADCASTHELP

(Broadcast Help) - Sent FROM E-MLE

Purpose EPM_BROADCASTHELP is sent a E-MLE application when the help instance has been initialized. To insure that all existing edit windows add their helpsub tables to the help instance EPM_QHELP_TABLE is broadcasted to all edit windows on the desktop informing them that help has been loaded. If no edit windows are running then EPM_EDIT_QUERY_HELP_INSTANCE is issued to the shell in order to create the help instance.

Message Parameter 1

not used (LONG)

Message Parameter 2

not used()

Return

NULL

Remarks: This is how EPM.EXE handles this messages.

```
case EPM_BROADCASTHELP:
   {
     USHORT rc;
     rc=ERESBroadcastMsgToEditWindows(GlobData, EPM_QHELP_TABLE,  L,
     if(!rc) {
        WinSendMsg(GlobData->eres.hwndAppClient,
                 EPM_EDIT_QUERY_HELP_INSTANCE,  L,  L);
     }
   }
   break;
```

EPM_EDIT_DELETEFILE

(Delete file from disk)

Purpose Delete the file specified in mp1 from disk and free pointer

Message Parameter 1

filename(PSZ) Fully qualified file name.

Message Parameter 2

not used()

Return

NULL

Remarks

:

EPM_EDIT_ACTIVATEFILEID

(Activate a file)

Purpose Activate a file corresponding to the specified file identifier.

Message Parameter 1

fileid (LONG) A file identifier that could be obtained using the EtkQueryFileID function.

Message Parameter 2

not used()

Return

NULL

Remarks:

EPM_EDIT_CHAR

(Issue a WM_CHAR message)

Purpose An alternitive to the WM_CHAR message. It is handled exactly in the same manner as WM_CHAR.

Message Parameter 1

mp1(MPARAM) See WM_CHAR in the PM Tech. Ref.

Message Parameter 2

mp2(MPARAM) See WM_CHAR in the PM Tech. Ref.

Return

Remarks:

EPM_EDIT_CLIPBOARDCOPY

(Insert text into the PM Clipboard)

Purpose Fast path to inserting text into the PM Clipboard.

Message Parameter 1

pTextBuf(PVOID) Pointer to a memory buffer containing text to copy to the clipboard. The text is in a format described by mp2.

Message Parameter 2

mp2() Flag that describes what format of the memory buffer, which was passed in mp1.

```
  = CF_TEXT type buffer, terminated by nul
1 = EPM shared memory buffer (32byte head)
```

Return

Remarks: When the contents of mp1 is copied to the clipboard a EPM defc event is called by the name of PROCESSCLIPBOARDCOPY. Arg(1) of this function is the original buffer passed in as mp1. The caller may choose to free the buffer during this command. If zero is passed as arg(1), an error was encountered. An error message should be displayed at this point.

EPM_EDIT_COMMAND

(Editor Command Message - E-MLE)

Purpose Issue a command to the E-MLE

Message Parameter 1

Edit command (PSZ) Any legal editor command. (See Appendix B for a complete list of commands)

Message Parameter 2

command flags(USHORT) The following command flags can be set with the EPM_EDIT_COMMAND message:

- COMMAND_SYNC - the command is sent to the edit interpreter thread and is executed immediately or after the last command.
- COMMAND_FREESEL - free the selector of parameter one when done.
- COMMAND_GETABLE - the selector in parameter one is a getable shared segment.

Returns

void

Remarks This message is the main source of communication to editor windows. To send a message to a specific file, send any legal editor command (in string form) to the associated edit window handle. Legal editor commands consist of any of the base editor commands or any user defined commands. User commands can be created within the editor macro language via DEFC's.

For command return codes see the ***EPM_EDIT_RETCODE*** message.

See Appendix B for a complete list of commands

By default all commands sent or posted to edit windows are copied and posted to the editor interpreter thread. By setting the COMMAND_SYNC flag the command is sent to the editor interpreter, therefore it will be executed immediately or after the last command is finished executing.

When an application other then the one that created an edit window wishes to send a command to an edit window, that application must allocate a shared segment for the command string (parameter one). The COMMAND_GETABLE flag notifies the E Toolkit that the pointer in parameter one is that of a getable shared segment.

When an application wants the selector sent in parameter one to be freed after it is used by the E Toolkit, the COMMAND_FREESEL flag must be set. This flag is only valid for selectors allocated within the application that created the edit window. A selector of a getable shared segment can't be freed by the E Toolkit with this message.

EPM_EDIT_COMMAND2

(Execute Command) - Sent TO an E-MLE window

Purpose Same as EPM_EDIT_COMMAND, but if the command does not exist, no error message is generated.

Message Parameter 1

See EPM_EDIT_COMMAND

Message Parameter 2

See EPM_EDIT_COMMAND

Return

See EPM_EDIT_COMMAND

Remarks

EPM_EDIT_CHANGEFONT

(Change E-MLE's Font)

Purpose Change E-MLE's base font.

Message Parameter 1

name(PSZ)
Font Face Name.

Message Parameter 2

size(USHORT)
Point Size.

style(USHORT)
Attribute Style.

Remarks

EPM_EDIT_TASKLIST

(Show/Hide Task list enty)

Purpose This message can either show or hide a task list entry corresponding to an E-MLE window. It also allows an application to set a prefix string before the task list entry, which typically contains the name of the visible file.

Message Parameter 1

flag(LOUSHORT)
Show task list entry flag
TRUE=Show entry
FALSE = Hide entry

Message Parameter 2

Prefix_string(PSZ)
String to be pre-pended to the task list entry

Remarks This message can be either sent of posted, however, if this message is posted, the application is responsible for freeing the string.

EPM_EDIT_RETCODE

(Editor Command Message Return Code)

Purpose Send to the registered owner in response to a edit command.

Message Parameter 1

Return string (PSZ)
Text describing the return code.

Message Parameter 2

return code (USHORT)
specific editor return code.

Remarks

Sent to owner registered in the Editor Information structure. (See Appendix A for more details on the Edit Information structure.) This message is in response to a internal editor return code/ error code.

See Appendix C for editor return codes.

EPM_EDIT_CURSORMOVE

(Editor Cursor Moved)

Purpose The cursor has changed position in the edit window.

Message Parameter 1

row (ULONG)
row of cursor in file

Message Parameter 2

column (ULONG)
column of cursor in file

Remarks

Sent to owner registered in the Editor Information structure. (See Appendix A for more details on the Edit Information structure.) This message gives the owner the capability to track the editor's cursor.

EPM_EDIT_ACTIVEHWND

(Editor received focus)

Purpose Notify application that a specific edit window has become active.

Message Parameter 1

Edit Window Handle (HWND)
Handle of active edit window.

Message Parameter 2

active file name (PSZ)
Full path of top most file being edited in active edit window.

Remarks

Processing this message is useful if more than one edit window is being used by a given application. The application does not need to keep global information on each edit window because it is returned via this message.

EPM_EDIT_OPTIONS

(Get Editor Option Information) - Sent TO E-MLE

Purpose Get a specified piece of editor information.

Message Parameter 1

Option Number (LONG)

OPTIONS_xxxxx constant. (See Remarks.)

Message Parameter 2

[Buffer](PVOID)

Used by some options.

Return The returned value depends on the option selected.

Constant that specifies one of the following options:

OPTIONS_MARGINS Returns 4 bytes containing (lo to hi): left marg (CHAR), right marg (CHAR), paragraph margin (CHAR), 0 (CHAR).

OPTIONS_LINE Returns line of file that contains the cursor. (ULONG)

OPTIONS_COLUMN Returns column of file that contains the cursor. (ULONG)

OPTIONS_INSERT Returns TRUE if cursor is in insert mode. (ULONG)

OPTIONS_AUTOSAVE Returns the number of changes before autosave. (ULONG)

OPTIONS_NTABS Returns the number of tab stops. (ULONG)

OPTIONS_NROWS Returns the total number of rows visible in the E-MLE. (uses the default font height as a measument.)

OPTIONS_NCOLS Returns the total number of columns visible in the E-MLE. (uses the default font width as a measument.) (ULONG)

OPTIONS_MODIFY Returns TRUE if the file has been modified. (last save) (ULONG)

OPTIONS_TAB Returns a pointer to a byte-table of tab values. The table is terminated by a NULL character. (PBYTE)

OPTIONS_SEARCH Returns the last string searched by locate or change. (PSZ)

OPTIONS_GETTEXT Returns the line of text specified by high word of option parameter. If high word is NULL the the line in which the cursor is currently on is returned. Parameter 2 must contain a pointer to a 255 byte character array. The specified line will be placed into this buffer. (PSZ)

OPTIONS_NAME Returns the name of the visible file.

OPTIONS_HWNDEXTRA Returns the handle to the extra area window. (HWND)

OPTIONS_HWNDEIOBJECT Returns the handle to the EI Work Thread. (PVOID)

OPTIONS_TEXTCOLOR Returns the text Foreground (LOUSHORT) and Background (HIUSHORT).

OPTIONS_RING Returns a buffer filled with the name of the files within the E-MLE ring. The buffer has the following format:

```
USHORT - length of filenames
UCHAR  - delimitor
ULONG  - fileid
CHAR[] - delimited string
```

OPTIONS_FILEID Returns the File ID of the visible file.

OPTIONS_QSELECTION Returns the type of selection.

This message is send to a edit window to get a specified option. All options are returned in the form of a 4 byte return code. Here is an example of using the Option message to get the total numbers of visible lines in a edit window.

```
VisibleLines = (ULONG)WinSendMsg( hwndEditWindow,
                                  EPM_EDIT_OPTIONS,
                                  OPTIONS_NROWS,
                                   L
                                 );
```

EPM_EDIT_ID

Query Editor ID number

Purpose Determine if a window handle is your E-MLE window.

Message Parameter 1

NULL
not used

Message Parameter 2

NULL
not used

Remarks

This message is used to determine if a window handle is a edit window handle. This message returns a 4byte return code. The way to determine if the specified handle is a edit window is by checking if the Hi-order word contains the value of EPM_EDIT_ID and the Lo-order word contains the value specified in the create structure field, 'editid'. If a match is found in the Hi-order word then the window handle is a edit window. If a match is found in both the Hi-order word and the Lo-order word, then the window handle is a edit window created by your application. An example follows:

```
result = WinSendMsg( hwndE, EPM_EDIT_ID,  L,  L );
match  = MPFROM2SHORT(epm.editid,EPM_EDIT_ID);

if (result==match) {
   /* Edit window found */
}
```

EPM_EDIT_SHOW

(Make edit window visible/invisible)

Purpose Refresh and Show/Hide the edit window specified.

Message Parameter 1

Show Flag(Bool)
TRUE = Show and Refresh edit window
FALSE = Hide edit window

Message Parameter 2

NULL
not used

Remarks

When an edit window is created (using EtkCreate) the WS_VISIBLE flag is not used as a window style. Therefore, the edit window is not visible upon creation. It is recommended that an EPM_EDIT_SHOW message be sent to the edit window to be displayed instead of WinShowWindow(...). The difference between sending the EPM_EDIT_SHOW verses the WinShowWindow is that the EPM_EDIT_SHOW forces a window repaint.

EPM_EDIT_NEWFILE

(Open a new edit window)

Purpose Open a new edit window with the specified file.

Message Parameter 1

file name(PSZ)
Asciiz string pointer to any legal file name. (Wildcards are supported)

Message Parameter 2

string flags (MPARAM)
Flags specifying how to handle the pointer in parameter one.
The following are flags that can be set.

- COMMAND_GETABLE - The pointer is a getable shared segment. Use DosGetSeg(SELECTOROF(mp1)) to access the pointer.
- COMMAND_FREESEL - The selector must be freed after it is used by the application. Use DosFreeSeg(FP_SEG(mp1)) to free the selector.

Returns

void

Remarks

An edit window is requesting that another edit window be opened. This message should be handled by calling EtkCreate.: The macros used by the EPM editor use this message to implement the 'OPEN' command.

The Flags in parameter two are checked by anding parameter two with the constants COMMAND_GETABLE and COMMAND_FREESEL. For example to check if the COMMAND_GETABLE flag is set do the folowing:

```
if( LONGFROMMP(mp2) & COMMAND_FREESEL ) {
  /* the free selector flag is set */
  }
```

EPM_EDIT_DESTROYNOTIFY

(A E-MLE has been closed)

Purpose Notify owner that an E-MLE window has been closed.

Message Parameter 1

edit handle(HWND)
Handle to the client window of the E-MLE window that was just closed.

Message Parameter 2

message queue(HMQ) Handle to the message queue formally being used by the E-MLE window that was just closed.

Returns

void

Remarks

Sent to owner registered in the Editor Information structure. (See Appendix A for more details on the Edit Information structure.) This message gives the owner the ability to take some action when a edit window is closed. Note that the window handle passed is not valid. It is to be used for reference only.

EPM_EXTRAWINDOW_REFRESH

(Refresh Extra Window Control)

Purpose Repaint the optional editor extra window, updating all values.

Message Parameter 1

NULL(MPARAM)
not used

Message Parameter 2

NULL (MPARAM)
not used

Returns

VOID

Remarks

The Extra Window control can provide two extra lines of text, the status area and the message area.**:** To obtain the extra window handle, send a EPM_EDIT_OPTION message to the edit window in question. When sending the option message specify OPTION_HWNDEXTRA as message parameter one.

EPM_EDIT_CONTROLTOGGLE

(Toggle or Query the status of a base edit window control)

Purpose Toggle a editor control on/off or force the control on/off.

Message Parameter 1

Id(LOUSHORT)
Control identifier; See table below.

rce ON/OFF(HIUSHORT)
Set to 2=ON, or 1=OFF

Message Parameter 2

Query flag (MPARAM)
TRUE if control is to be Queried, otherwise FALSE.

Returns

void

Remarks

This message is send to an edit window to toggle (turn ON or OFF) some editor control window. The message can also be used to force a particular control on or off.**:** The lo-ordered short of the first message parameter contains an id number that determines the control window. The id constants are as follows:

EDITSTATUSAREA Status area of extra window control.
EDITMSGAREA Message area of extra window control
EDITVSCROLL Vertical scroll bar
EDITHSCROLL Horizontal scroll bar
EDITTITLEBAR Stops automatic updating of title bar with filename. This is useful if the application requires the title bar to display text other then the current file being edited.
EDITCURSOR Editor cursor

The hi-ordered short of the first message parameter contains an optional value. If the value is null then the control is toggled. If it is non- zero the control status is set to one less the value. This value should be set to 2 if you want to force the control on, or one if you want to force the control off.

Parameter contain the Query Flag. If parameter is non-zero the status of the specified control is returned. That is, TRUE is returned if the control is active or FALSE if it is not.

EPM_EDIT_RECORDKEY-

(Start recording keystrokes)

Purpose Start the recording of keystrokes (WM_CHAR messages) sent to the edit window.

Message Parameter 1

NULL (MPARAM)
not used

Message Parameter 2

NULL (MPARAM)
not used

Returns

void

Remarks

EPM_EDIT_ENDRECORDKEY-

(Stop storing keystrokes)

Purpose Terminates the storing of keystrokes (WM_CHAR messages) started by the EPM_EDIT_RECORDKEY message.

Message Parameter 1

NULL (MPARAM)
not used

Message Parameter 2

NULL (MPARAM)
not used

Returns

void

Remarks

EPM_EDIT_PLAYKEY

(Execute stored keystrokes) - Sent TO an E-MLE

Purpose Generate keystrokes (WM_CHAR messages) saved since the EPM_EDIT_RECORDKEY message has been sent.

Message Parameter 1

NULL (MPARAM)
not used

Message Parameter 2

NULL (MPARAM)
not used

Returns

void

Remarks

It is recommend that a EPM_EDIT_ENDRECORDKEY message be sent before this message is sent. If the edit window is still in record mode when the message is sent an error message is sent to the owner window.

EPM_EDIT_QUERYRECORDKEY

(Determine record key state)

Purpose Returns TRUE if edit window is recording keys.

Message Parameter 1

NULL (MPARAM)
not used

Message Parameter 2

NULL (MPARAM)
not used

Returns

TRUE = edit window is recording keys.

Remarks

EPM_EDIT_ASKTOQUIT

(Request to quit a modified file)

Purpose Notify application that a request has been sent to quit a file which has been modified.

Message Parameter 1

filename (PSZ)
Name of file that has been modified.

Message Parameter 2

heditwnd (HWND)
The edit window's handle.

Remarks

This message is sent after the E toolkit has received notice to quit a modified file. The application must respond to this message by sending an EPM_EDIT_ASKTODONE message. When an application receives this message it is a good time to pop a dialog to the user asking what action to take. (See the Quitbox dialog for an example)

Related Messages

```
EPM_EDIT_ASKTOCLOSE
EPM_EDIT_ASKTOFAILED
EPM_EDIT_ASKTODONE
```

EPM_EDIT_ASKTOCLOSE

(Request to close an edit window)

Purpose Notify application that a request has be sent to close an edit window which contains a file that has been modified.

Message Parameter 1

filename (PSZ)
Name of file that has been modified.

Message Parameter 2

heditwnd (HWND)
The edit window's handle.

Remarks

This message is sent after the E toolkit has received notice to close an edit window which contains a modified file. The application must respond to this message by sending an EPM_EDIT_ASKKTODONE message.

Related Messages

```
EPM_EDIT_ASKTOQUIT
EPM_EDIT_ASKTOFAILED
EPM_EDIT_ASKTODONE
```

EPM_EDIT_ASKTODONE

(Respond to EPM_EDIT_ASKTOxxx message)

Purpose Respond to one of the EPM_EDIT_ASKTOxxx messages from the E toolkit.

Message Parameter 1

retcode (MPFROMSHORT)
Response message sent to E toolkit. One of the following messages can be used:

- ERES_CANCEL - Cancel the closing of the file or edit window.
- ERES_DISCARD - Discard the modifications and continue closing.
- ERES_SAVE - Save file as message parameter two and continue closing.

Message Parameter 2

filename (PSZ) Name of file to save modifications in if EPM_SAVE is sent. This pointer must be the same pointer sent to the applications with one of the EPM_EDIT_ASKTOxxx messages.

Remarks

This message allows the application control the saving or discarding of a modified file when an edit window is closing, a quit file message has been sent, or when a previous EPM_EDIT_ASKTODONE message returned EPM_EDIT_ASKTOFAILED.

Related Messages

```
EPM_EDIT_ASKTQUIT
EPM_EDIT_ASKTOCLOSE
EPM_EDIT_ASKTODONE
```

EPM_EDIT_ASKTOFAILED

(Last EPM_EDIT_ASKTODONE failed)

Purpose Notify application that the file name sent with the last EPM_EDIT_ASKTODONE message could not be saved with the name specified.

Message Parameter 1

filename (filename)
File name used which caused an error in saving.

Message Parameter 2

heditwnd (HWND)
The edit window's handle.

Remarks

This message informs the application that the file name sent with the EPM_EDIT_ASKTODONE message and message parameter EPM_SAVE, was a bad file name. The application should correct the file name and send another EPM_EDIT_ASKTODONE message.

Related Messages

```
EPM_EDIT_ASKTQUIT
EPM_EDIT_ASKTOCLOSE
EPM_EDIT_ASKTOFAILED
```

ERES.DLL

ERES.DLL exports a set a general purpose functions that can be used by an application that contains E-MLE's. These functions are detailed in this section. ***In order to use these functions an application must do the following in the order given:***

1) Include ETOOLKT.H in all source files that use ERES.DLL functions. The main source file of the application should define the constant INCL_DONT_EXT_GLOB before including etoolkt.h. For example:

```
#define INCL_DONT_EXT_GLOB
#include <ETOOLKT.H>
```

2) Declare a variable of type GLOBDATA in the main source file of the application.

3) Load the ERES.DLL module and initialize the "eresModule" field of the GLOBDATA variable with the ERES.DLL module handle.

```
// Note that GlobData is a variable of type GLOBDATA
DosLoadModule((PSZ)NULL,(USHORT) ,(PSZ)"ERES",
            (PHMODULE)&GlobData.eresModule );
```

4) The following three function calls must be called next.

```
ERESRegisterEtkFunctions((PFN) EtkCreate,  CREATE_FUNC);
ERESRegisterEtkFunctions((PFN) EtkDestroy, DESTROY_FUNC);
ERESRegisterEtkFunctions((PFN) EtkVersion,      VERSION_FUN
```

(See ERESRegisterEtkFunctions in the section "ERES.DLL Exported Functions" for an explanation of these function calls)

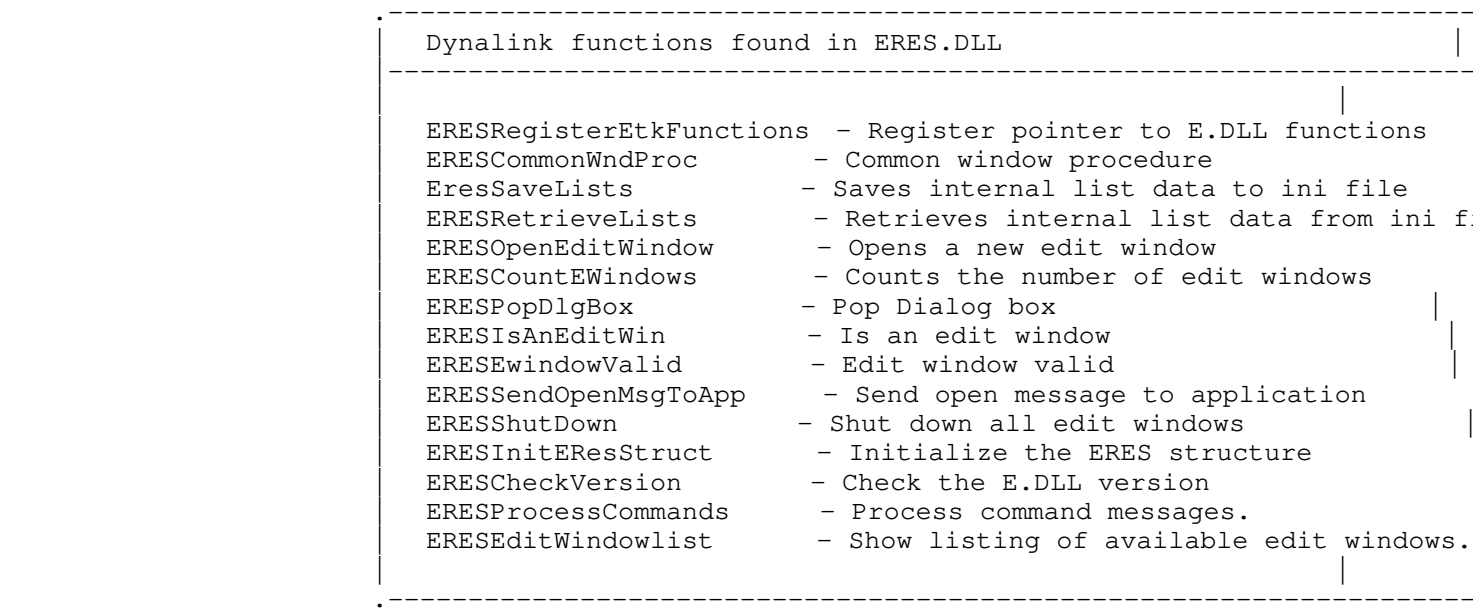
5) Call the function ERESInitEresGlob() in order to initialize the internal data of ERES.DLL.

6) Required fields of the variable of type GLOBDATA must be initialized. These are:

- eres.exfilename - name of the applications EX file.
- ApplicationName - name of the application.
- ExSearchPath - search path for EX files. Set to NULL if search path includes only the current directory.
- ApplicationID - a unique ID that distinguishes the edit windows of the application.

ERES.DLL - Exported Functions

The following functions are exported from ERES.DLL



ERESRegisterEtkFunctions - (Registers pointers to E.DLL functions)

Allows ERES.DLL to call E.DLL functions without having to import them.

ERESRegisterEtkFunctions (,)

Parameters

FuncPtr *(PFN)* - input
Pointer to E.DLL function. One of the following can be used:

- EtkCreate
- EtkDestroy
- EtkVersion

ID *(USHORT)* - input
Function identifier. One of the following can be used:

- CREATE_FUNC - EditWindowCreate
- DESTROY_FUNC - EditWindowDestroy
- VERSION_FUNC - EditVersion

Returns

Void

Remarks

Currently an application must send the pointers of E.DLL functions required by ERES.DLL in order to avoid reloading of the E.DLL module. Due to an a debug in the operating system if two modules in the same application import the same module, that module is reloaded. Therefore, this function avoids the costly reloading of the E.DLL module. Util this problem is resolved the following code must be placed after an application loades eres.dll:

Example

```
ERESRegisterEtkFunctions((PFN) EtkCreate,  CREATE_FUNC);
ERESRegisterEtkFunctions((PFN) EtkDestroy, DESTROY_FUNC);
ERESRegisterEtkFunctions((PFN) EtkVersion, VERSION_FUNC);
```

ERESCommonWndProc

(Common window procedure)

Purpose Handles default window message processing of E toolkit applications.

Prototype: *ERESCommonWndProc*

Parameters

hwnd (HWND)
The window handle of the application.

msg (HWND)
The window message.

mp1 (MPARAM)
Message parameter one.

mp2 (MPARAM)
Message parameter two.

Returns
VOID

Remarks

Applications that choose to call this function as their default window procedure will have the default message processing for E toolkit messages. (See Appendix H for a description of these messages) The Window ULONG of the applications window handle must be set to the address of the GLOBDATA structure in order to process application specific information.

ERESSaveLists

(Saves internal list data to ini file)

Purpose Saves the internal data structures of the Open File, List Box, and Command Dialog boxes in the ini file specified in the GLOBDATA structure.

Prototype: *ERESSaveLists*

Parameters

GlobData (GLOBDATA *)
Pointer to application instance data.

Returns

Void

Remarks

GlobData->IniApplicationName contains the name of the ini file to store the structures in. GlobData->eres contains the structures cmd, msg, and open1 which are the structures used by the three dialog boxes.

ERESRetrieveLists

(Retrieves internal list data from ini file)

Purpose Retrieves the internal data structures of the Open File, List Box, and Command Dialog boxes from the ini file specified in the GLOBDATA structure.

Prototype: *ERESRetrieveLists*

Parameters

GlobData (GLOBDATA *)
Pointer to application instance data.

Returns

VOID

Remarks

GlobData->IniApplicationName contains the name of the ini file to retrieve the structures from. GlobData->eres contains the structures cmd, msg, and open1 which are the structures used by the three dialog boxes. These will be filled with the data in the specified ini file.

ERESOpenEditWindow

(Opens a new edit window)

Purpose Initializes the EDITORINFO structure with application specific data and calls the EPM_CreateEditWindow function of E.DLL .

Prototype: *ERESOpenEditWindow*

Parameters

filename (char far *)
Pointer to the string containing the file to edit.

GlobData (GLOBDATA *)
Application instance data.

Returns ***hwndEdit*** (HWND) The window handle of the new edit window.

Remarks

ERESCountEWindows

(Counts the number of E-Multi-line Edit windows)

Purpose Determines the number of available edit windows.

Prototype: *ERESCountEWindows*

Parameters

GlobData (GLOBDATA *) Pointer to application instance data.

Returns ***count*** (USHORT) The total number of E-Multi-line edit windows that belong to an application.

Remarks

GlobData->CountOnlyVisibleWindows determines whether to count only visible windows. Some applications might create edit windows, keep them hidden, and use them at some later time.

ERESPopDlgBox

(Pop Dialog Box)

Purpose Display the specified dialog box. If the dialog is already being displayed bring it to the top.

Prototype: *ERESPopDlgBox(*

Parameters

proc (PFNWP) Pointer to the window procedure of the dialog.

dlgin (USHORT) Pointer to dialog input information.

ModuleType (PSZ) Specifies the module that contains the dialogs resource information excluding the ".DLL" extension. If set to NULL then the dialog resource is contained in ERES.DLL.

ownertype (USHORT) The owner of the dialog. This can be set to one of four values:

- OT_DEFAULT - Checks the value of the active edit window. If it is valid make it the owner, else make the handle to the applications frame the owner.
- OT_ACTIVE_EDIT - Make the active edit window the owner.
- OT_APP_FRAME - Make the applications frame the owner.
- OT_DESKTOP - Make the desktop window the owner.

GlobData (GLOBDATA *) Application instance data.

Returns

VOID

Remarks

ERESIsAnEditWin

(Is an edit Window)

Purpose Test if a given window is an edit window.

Prototype: *ERESIsAnEditWin*

Parameters

hwnd (HWND) Window handle of the window to test.

GlobData (GLOBDATA *) Pointer to application instance data.

Returns *rc*(BOOL)

- TRUE - The window is an edit window.
- FLASE - The window is not an edit window.

Remarks

ERESEwindowValid

(Is an Edit Window valid)

Purpose Tests whether the active edit window of the application instance data is valid.

Prototype: *ERESEwindowValid*

Parameters

GlobData (GLOBDATA *) Pointer to application instance data.

Returns *rc*(BOOL)

- TRUE - The the active edit window handle is valid.
- FLASE - The the active edit window handle is not valid.

Remarks

ERESSendOpenMsgToApp

(Send open message to application)

Purpose Finds the handle of applications client window, and posts it a message to open the files in the shared message buffer of the application instance data.

Prototype: *ERESSendOpenMsgToApp*

Parameters

ID (USHORT) Message response to sending the EPM_BOOK_ID message to the applications window.

Returns

VOID

Remarks

This function traverses through all existing frame windows searching for the applications window. The Custom message EPM_BOOK_ID is used to test whether a windows is that of the application. If a window responds to an EPM_BOOK_ID message with ID then it is the applications window. GlobData->ShrMsgBuff contains the name of the file to open.

ERESShutDown

(Shut Down)

Purpose Systematically closes all E-MLE's associated with an application.

Prototype: *ERESShutDown*

Parameters

GlobData (GLOBDATA *) Pointer to application instance data.

Returns *rc*(BOOL)

- TRUE - A given file in an edit window does not wish to close down because it has yet to be saved.
- FLASE - Shut down of all edit windows completed.

Remarks

ERESInitEResStruct

(Initialize the ERES structure)

Purpose Initializes the eres field of the GlobData structure.

Prototype: *ERESInitEResStruct*

Parameters

hwnd (HWND) The window handle of the applications client window.

GlobData (GLOBDATA *) Pointer to applications instance data.

Returns

VOID

Remarks

The pointer to this structure is copied and used internally in ERES.DLL.

ERESCheckVersion

(Check ERES.DLL version)

Purpose Compare the ERES.DLL's version number with that of E.DLL .

Prototype: *ERESCheckVersion*

Parameters

No Parameters

Returns *rc* (BOOL)

- TRUE - The Version numbers are the same.
- FLASE - Version conflict exits.

ERESProcessCommands

(Process command messages)

Purpose Process WM_COMMAND messages sent to the applications default window procedure.

Prototype: *ERESProcessCommands*

Parameters

mp1 (MPARAM) WM_COMMAND message parameter.

mp2 (MPARAM) Pointer to dialog input information.

GlobData (GLOBDATA *) Pointer to application instance data.

Returns

VOID

Remarks

This functions is used when ERESCommonWndProc() is the default window procedure of the application.

ERESEditwindowList

(Show listing of available edit windows)

Purpose Create a Popup menu that contains the names of the available edit windows which the user can transfer control to.

Prototype: *ERESEditwindowList*

Parameters

GlobData (GLOBDATA *) Pointer to application instance data.

Returns

VOID

Remarks

GlobData->EditWindowListStyle controls the positioning of the edit list window dialog relative to the applications client window. (See Appendix F for the available options)

ERES.DLL - Exported Dialog Box Procedures

The following dialog box procedures are exported by ERES.DLL. In order to use any of these dialogs an application must adhere to the required code outlined in the previous section.

- ***ERESOpenDlgProc*** Open dialog procedure. Full file selection dialog box includes fields specifying drives, directories, files, path name, and a file name entry field.
- ***ERESOpen1DlgProc*** Open dialog procedure. Contains a list of previously entered file names, and an entry field.
- ***ERESCommandDlgProc*** Edit window command dialog box.
- ***FindChangeDlgProc*** Allows find/change commands to be enter from dialog box.
- ***ConfigDlgProc*** Configuration edit window dialog procedure.
- ***EntryBoxDlgProc*** General purpose entry dialog box.
- ***ListBoxDlgProc*** List box dialog.
- ***QuitBoxDlgProc***
  Quit box dialog.

ERESOpen1DlgProc

(Open dialog procedure one)

Purpose Open1 dialog box procedure.

Prototype: ***ERESOpen1DlgProc***

Parameters

hwnd (HWND)
The handle to the dialog box.

message (USHORT)
Window message.

lparm1 (MPARAM)
Message parameter one.

lparm2 (MPARAM)
Message parameter two.

Returns

VOID

Remarks

This dialog contains a list of previously entered file names, and an entry field. This dialog can be called using the WinDlgBox() function. For example:

```
WinDlgBox(Parent, Owner, DlgProc, Resource, Dlgid, CreatParam);
```

- Parent - Window handle of the parent of the dialog
- Owner - Window handle of the owner of the dialog
- DlgProc - Window procedure of the dialog (ERESOpen1DlgProc)
- Resource - Where the dialog resource is located (ERES)
- Dlgid - Dialog id (DLG_OPEN1)
- CreateParam - Dialog creation parameter. Must be set to(&GlobData->eres).

ERESCommandDlgProc

(Command dialog procedure)

Purpose Edit window command line dialog procedure.

Prototype: *ERESCommandDlgProc*

Parameters

hwnd (HWND)
The handle to the dialog box.

message (USHORT)
Window message.

lparm1 (MPARAM)
Message parameter one.

lparm2 (MPARAM)
Message parameter two.

Returns

VOID

Remarks

This is an edit window command line dialog box, which includes an entry field and a scrollable list of previous commands. This dialog can be called using the WinDlgBox() function. For example:

```
WinDlgBox(Parent, Owner, DlgProc, Resource, Dlgid, CreatParam);
```

- Parent - Window handle of the parent of the dialog
- Owner - Window handle of the owner of the dialog
- DlgProc - Window procedure of the dialog (ERESCommandDlgProc)
- Resource - Where the dialog resource is located (ERES)
- Dlgid - Dialog id (IDM_COMMANDS)
- CreateParam - Dialog creation parameter. Must be set to(&GlobData->eres).

ERESFindChangeDlgProc

(Find/Change dialog procedure)

Purpose Find/Change commands dialog procedure.

Prototype: ***ERESFindChangeDlgProc***

Parameters

hwnd (HWND)
The handle to the dialog box.

message (USHORT)
Window message.

lparm1 (MPARAM)
Message parameter one.

lparm2 (MPARAM)
Message parameter two.

Returns

VOID

Remarks

This dialog allows find/change commands to be entered directly from it. This dialog can be called using the WinDlgBox() function. For example:

```
WinDlgBox(Parent, Owner, DlgProc, Resource, Dlgid, CreatParam);
```

- Parent - Window handle of the parent of the dialog
- Owner - Window handle of the owner of the dialog
- DlgProc - Window procedure of the dialog (ERESFindChangeDlgProc)
- Resource - Where the dialog resource is located (ERES)
- Dlgid - Dialog id (DLG_CHANGE).
- CreateParam - Dialog creation parameter. Must be set to(&GlobData->eres).

ERESConfigDlgProc

(Configuation dialog procedure)

Purpose Allow edit window options to be configured from dialog box.

Prototype: *ERESConfigDlgProc*

Parameters

hwnd (HWND)
The handle to the dialog box.

message (USHORT)
Window message.

lparm1 (MPARAM)
Message parameter one.

lparm2 (MPARAM)
Message parameter two.

Returns

VOID

Remarks

This dialog allows configuration of edit window options. These options include setting margins, autosave level, tabs, colors, and paths. This dialog can be called using the WinDlgBox() function. For example:

```
WinDlgBox(Parent, Owner, DlgProc, Resource, Dlgid, CreatParam);
```

- Parent - Window handle of the parent of the dialog
- Owner - Window handle of the owner of the dialog
- DlgProc - Window procedure of the dialog (ERESConfigDlgProc)
- Resource - Where the dialog resource is located (ERES)
- Dlgid - Dialog id (DLG_CONFIG)
- CreateParam - Dialog creation parameter. Must be set to(&GlobData->eres).

EntryBoxDlgProc

(Entry dialog box procedure)

Purpose General purpose entry dialog box.

Prototype: *EntryBoxDlgProc*

Parameters

hwnd (HWND)
The handle to the dialog box.

message (USHORT)
Window message.

lparm1 (MPARAM)
Message parameter one.

lparm2 (MPARAM)
Message parameter two.

Returns

VOID

Remarks

This dialog can be called using the WinDlgBox() function. For example:

```
WinDlgBox(Parent, Owner, DlgProc, Resource, Dlgid, CreatParam);
```

- Parent - Window handle of the parent of the dialog
- Owner - Window handle of the owner of the dialog
- DlgProc - Window procedure of the dialog (EntryBoxDlgProc)
- Resource - Where the dialog resource is located (ERES)
- Dlgid - Dialog id (IDD_ENTRYBOX)
- CreateParam - Dialog creation parameter. Must be set to(PENTRYBOXINFO). Check eres.h for a desription of the structure ENTRYBOXINFO. A pointer to this structure must be allocated and passed into the dialog procedure via this parameter.

ListBoxDlgProc

(List box dialog procedure)

Purpose General purpose list dialog procedure.

Prototype: *ListBoxDlgProc*

Parameters

hwnd (HWND)
The handle to the dialog box.

message (USHORT)
Window message.

lparm1 (MPARAM)
Message parameter one.

lparm2 (MPARAM)
Message parameter two.

Returns

VOID

Remarks

This dialog can be called using the WinDlgBox() function. For example:

```
WinDlgBox(Parent, Owner, DlgProc, Resource, Dlgid, CreatParam);
```

- Parent - Window handle of the parent of the dialog
- Owner - Window handle of the owner of the dialog
- DlgProc - Window procedure of the dialog (ListBoxDlgProc)
- Resource - Where the dialog resource is located (ERES)
- Dlgid - Dialog id (ID_LISTBOX)
- CreateParam - Dialog creation parameter. Must be set to(PLISTBOXINFO). Check eres.h for a description of the LISTBOXINFO structure. A pointer to this structure must be allocated and passed into the dialog procedure via this parameter.

QuitBoxDlgProc

(Quit box dialog procedure)

Purpose Sample Quit box dialog which handles EPM_EDIT_ASKTOxxx messages.

Prototype: *QuitBoxDlgProc*

Parameters

hwnd (HWND)
The handle to the dialog box.

message (USHORT)
Window message.

lparm1 (MPARAM)
Message parameter one.

lparm2 (MPARAM)
Message parameter two.

Returns
VOID

Remarks

This dialog informs the user that a file has been modified. It includes an entry field to specify what name the modifications should be saved as, and the pushbuttons: Save , Discard, Cancel, and Help. This dialog can be called using the WinDlgBox() function. For example:

```
WinDlgBox(Parent, Owner, DlgProc, Resource, Dlgid, CreatParam);
```

- Parent - Window handle of the parent of the dialog
- Owner - Window handle of the owner of the dialog
- DlgProc - Window procedure of the dialog (QuitBoxDlgProc)
- Resource - Where the dialog resource is located (ERES)
- Dlgid - Dialog id (IDD_QUITBOX)
- CreateParam - Dialog creation parameter. Must be set to(PQUITBOXINFO). See eres.h for a description of the QUITBOXINFO structure. A pointer to this structure must be allocated and passed into the dialog via this parameter.

Returns

This dialog returns one of the following codes.

- ERES_SAVE - Save was selected. File name is stored in QUITBOXINFO->filename
- ERES_CANCEL - Cancel was selected.
- ERES_DISCARD - Discard was selected.

Example E Toolkit Application

The following sample source code demonstrates how an application can use various features of the E Toolkit. The source files included are:

- Edllsamp.c - creates a sample edit window using E.DLL
- edllsamp.e - macros defining the edit windows profile
- eressamp.c - dialog resource management functions
- sampdlg.c - sample dll which demonstrates two types of PM dialogs.
- sampdlg.e - macros demonstrating how to invoke the sampdlg.c dialogs.

Appendix

Appendix A -

Description of the Editor Information Structure. (EDITORINFO)

```
/*-------------------------------------------------------------------*
 |  Editor Information Structure defined in 'C'                      |
 |-------------------------------------------------------------------|
 | typedef struct EDIT_INFO_TYPE {                                   |
 |   HAB      hab;          /* application anchor block           */|
 |   HWND     hwndparent;    /* handle to parent of edit window     |
 |   HWND     hwndowner;     /* handle to owner of edit window      |
 |   PRECTL    rect;         /* positioning of edit window         */|
 |   PSZ      filename;      /* file to be edited (with wildcard)   |
 |   HPOINTER  hEditPtr;      /* handle to editor pointer icon.     |
 |   HPOINTER  hMarkPtr;      /* handle to mark pointer icon.       |
 |   VOID FAR *hEditorIcon;   /* editor ICON.                      */|
 |   ULONG    editorstyle;   /* internal editor options             */
 |   ULONG    pmstyle;       /* PM standard window styles (FCF_xxxx)
 |   USHORT    font;         /* TRUE = LARGE FONT,  FALSE = SMALL FONT*
 |   PSZ      exfile;        /* pre-compiled macro code file (xxx.EX)
 |   PSZ      *topmkr;       /* top and bottom of file marker       */
 |   PSZ      *botmkr;       /*                                   */|
 |   SHORT    editid;        /* unique window id specified for  window
 |   PSZ      exsearchpath;  /* environment varible-search for .ex fi
 |   SHORT    reserved ;     /* for future use.                    */|
 | } EDITORINFO;                                                     |
 |                                                                   |
  -------------------------------------------------------------------*/
```

hab Application Anchor Block. To obtain the applications anchor block use the WinInitialize function. (example: hab = WinInitialize(NULL);)

hwndparent Window handle of the edit window parent. Effects the positioning of the edit window. In EPM, for example, the parent to the edit window is the desk top window (HWND_DESKTOP). This is why the edit windows created by EPM are free to roam the entire desk top.

hwndowner Window handle of the edit window owner. Determines which window receives messages from the edit window. For example, if a command is send to the edit window (via the EPM_EDIT_COMMAND message), and the command generates some type of return code. The return code is put into message form (via the EPM_EDIT_RETCODE message), and passed to the edit window owner.

rect Pointer to a PM rectangle structure. Determines the positioning of the edit window. The '.xLeft' and '.yBottom' fields set the lower lefthand corner of the edit window. The '.xRight' and '.yTop' fields set the width and height of the edit window. Since this is a pointer to a rectangle structure, pass the address of a RECTL in the following manner: epm.rect = ▭

filename Pointer to a Asciiz string that contains a fully qualified path. If 'filename' is NULL then a blank file will be edited. The blank file will be named '.Unnamed file'. 'filename' can contain wildcards. For example, if '*.dat' were specified, all the files with the extension '.dat' will be placed in the newly created edit window.

hEditPtr Handle to a pointer icon that will become active when the mouse pointer is over the client area of the edit window. Use WinQueryPointer(HWND_DESKTOP); to use current mouse pointer.

hMarkPtr Handle to a pointer that will be displayed when editor is marking text.

hEditIcon Handle to a icon that will be displayed when the edit window is minimized.

editorstyle Internal Editor style flags. Logical 'OR' these flags together to form desired editor configuration.

- EDIT_STYLE_BROWSE - browse file (view file only)
- EDIT_STYLE_ACTIVEFOCUS - when edit window becomes active it will take focus.
- EDIT_STYLE_STATUSLINE - Create edit window with status line
- EDIT_STYLE_MESSAGELINE - Create edut window with message line

pmstyle

Standard PM window FCF_xxx styles. For example, if you want to create a edit window with a Title Bar, Vertical Scroll Bar, and Size Border, you would use the following flags:

```
FCF_TITLEBAR | FCF_SIZEBORDER | FCF_VERTSCROLL
```

nt Flag that specifies the size font to use in the edit window. If font is TRUE, the largest possible system AVIO font will be used. If font is FALSE, the smallest possible system AVIO font will be used.

exfile Pointer to a asciiz string containing the name of a .EX file. A .EX file contains compiled macro code. The file must be created using ETPM.EXE. If this parameter is NULL, "EPM.EX" is used.

topmkr Pointer to a asciiz string containing the text that will be used to denote the top of file.

botmkr Pointer to a asciiz string containing the text that will be used to denote the bottom of file.

editid A unique editor id number that is used to identify edit windows created by a particular application.

exsearchpath A pointer to a string that contains the name of an environment variable. The internal toolkit will use the paths associated with this environment variable to search for .ex files specified during the life of an edit window. If this field is NULL, EPMPATH is searched by default.

reserved0 Reserved for future enhancements.

Appendix A1 -

Field Constants

Used by EtkSetFileField(...), EtkQueryFileField, and EtkQueryFileFieldString.

```
ATTRIBUTE_SUPPORT_LEVEL  -     = No Attributes supported
AUTOSAVE_FIELD           -   File Modifies Before Autosaving
CODEPAGE_FIELD           -   Code page
COL_FIELD                -   Column of Cursor.
CURSORX_FIELD            -   X Cursor Position in pels.
CURSORYG_FIELD           -
CURSORY_FIELD            -   Y Cursor Position in pels.
CURSOR_COLUMN            -   column of the cursor
CURSOR_OFFSET            -
DRAGCOLOR_FIELD          -   drag shadow
DRAGSTYLE_FIELD          -   drag type (linbe, block, character)
EA_AREA_FIELD            -   pointer to extended attribute area
FILENAME_FIELD           -   file name
FONTHEIGHT_FIELD         -   average font width
FONTWIDTH_FIELD          -   average font height
FONT_FIELD               -   internal font identifier string
KEYSET_FIELD             -   Key set name
LAST_FIELD               -   Number of last line in file. (absolute pos
LINEG_FIELD              -
LINE_FIELD               -   Number of line where cursor is located. (a
LOCKHANDLE_FIELD         -   TRUE = File is locked
MARGINS_FIELD            -   Margin String. ( Left Right Paragraph )
MARKCOLOR_FIELD          -   Color of selected text
MODIFY_FIELD             -   number of file modifications
MOUSEX_FIELD             -   x position of mouse
MOUSEY_FIELD             -   y position of mouse
READONLY_FIELD           -   TRUE = BROWSE MODE
SCROLLX_FIELD            -   x scroll increment
SCROLLY_FIELD            -   y scroll increment
TABS_FIELD               -   Tab rack.
TEXTCOLOR_FIELD          -   forground and background text color
TITLETEXT_FIELD          -   text to be placed on title bar
USERSTRING_FIELD         -   string to be used by user
VISIBLE_FIELD            -   file visible flag (FALSE=NOT in ring)
WINDOWHEIGHTG_FIELD      -
WINDOWHEIGHT_FIELD       -   height of window in pels
WINDOWWIDTHG_FIELD       -
WINDOWWIDTH_FIELD        -   width of window in pels
WINDOWX_FIELD            -   starting x position of window
WINDOWY_FIELD            -   starting x position of window
```

Appendix B - Editor Commands.

See EPM Users guide for a list of editor Command. (EPM SCRIPT)

Appendix C -

Editor Return Codes

Descriptor Constant	Return Code Value
SEE_MESSAGE	
FILE_NOT_FOUND_RC	-2
PATH_NOT_FOUND_RC	-3
TOO_MANY_OPEN_FILES_RC	-4
ACCESS_DENIED_RC	-5
MEMORY_CONTROL_BLOCKS_RC	-7
INSUFFICIENT_MEMORY_RC	-8
INVALID_DRIVE_RC	-15
NO_MORE_FILES_RC	-18
NUMERIC_OVERFLOW_RC	-254
INVALID_NUMBER_ARGUMENT_RC	-255
RECURSION_TOO_DEEP_RC	-256
INVALID_NUMBER_OF_PARAMETERS_RC	-257
OUT_OF_STRING_SPACE_RC	-258
EXPRESSION_STACK_OVERFLOW_RC	-259
INVALID_FILEID_RC	-26
ILLEGAL_OPCODE_RC	-261
TOO_MANY_WINDOWS_RC	-262
INVALID_ARGUMENT_RC	-263
LOOP_STACK_OVERFLOW_RC	-264
DIVIDE_BY_ZERO_RC	-265
UNABLE_TO_SHRINK_RC	-266
INVALID_CALL_BY_REFERENCE_RC	-267
PROCEDURE_NEEDS_MORE_ARGUMENTS_	-268
BREAK_KEY_PRESSED_RC	-269
NOT_ENOUGH_MEMORY_RC	-27

ERROR_IN_MARGIN_SETTINGS_RC	-271
ERROR_IN_TAB_SETTINGS_RC	-272
STRING_NOT_FOUND_RC	-273
UNKNOWN_COMMAND_RC	-274
MISSING_FILENAME_RC	-275
LINE_TOO_LONG_TO_JOIN_RC	-276
TOO_MANY_FILES_RC	-277
LINES_TRUNCATED_RC	-278
TEXT_ALREADY_MARKED_RC	-279
TEXT_NOT_MARKED_RC	-28
SOURCE_DEST_CONFLICT_RC	-281
NEW_FILE_RC	-282
LINE_MARK_REQUIRED_RC	-283
ERROR_OPENING_FILE_RC	-284
ERROR_WRITING_FILE_RC	-285
ERROR_READING_FILE_RC	-286
INSUFFICIENT_DISK_SPACE_RC	-287
BLOCK_MARK_REQUIRED_RC	-288
TOO_MANY_RINGS_RC	-289
INCORRECT_VERSION_RC	-29
NO_MAIN_ENTRY_POINT_RC	-291
ERROR_CLOSING_FILE_RC	-292
CMDLINE_TOO_LONG_RC	-3
CANT_UNLINK_MOD_IN_USE_RC	-3 1
CANT_UNLINK_KEY_MOD_RC	-3 2
INTERNAL_INVALID_MOD_NBR_RC	-3 3
LINK_MODULE_ERROR_RC	-3 4

MAIN_NOT_FOUND_RC	-3 5
INIT_NOT_FOUND_RC	-3 6
LOADFILE_FINDFILE_RC	-3 7
LOADFILE_MAKEFILE_RC	-3 8
LOADFILE_ALREADYLINKED_RC	-3 9
UNLINK_UNKNOWN_MODULE_RC	-31
UNLINK_BAD_MODULE_FN_RC	-311
CALL_DUPLICATED_PROC_RC	-312
CALL_UNKNOWN_PROC_RC	-313
GREP_MEMORY_ERROR	-314
GREP_MISSING_BRACKET	-315
GREP_BAD_RANGE	-316
GREP_EMPTY_RANGE	-317
GREP_REGULAR_EXPRESSION_LONG	-318
DYNALINK_INCORRECT_PARAMETERS	-319
CANNOT_FIND_KEYSET	-321
BAD_LIBRARY_OR_PROC	-322
INVALID_LINE_NUMBER	-323
KBDSETSTATUS_FAILED	-324
BUFFER_CREATE_SIZE	-325
BAD_PROCEDURE	-326

Appendix E -

Description of the Global Data Structure. (GLOBDATA)

```
+-------------------------------------------------------------------
|   Global Data Structure defined in 'C'                           |
|-------------------------------------------------------------------
|   typedef struct {                                                |
|      HAB        hAB;                                             |
|      EPMRES      eres;                                           |
|      HMODULE     eresModule;                                      |
|      HMQ        hmq;                                             |
|      HDC        hDCApp;                                          |
|      PFNWP       DefFrameProc;                                   |
|      PSZ        ShrMsgBuff;                                      |
|      RECTL       rc;                                             |
|      USHORT      stagcount;                                      |
|      CHAR       LastCmdLine[256];                                |
|      CHAR       EPMClass[32];                                    |
|      CHAR       AllowMultProcesses;                              |
|      HWND       hwndHelpInstance;                                |
|      HELPINIT    hmiHelpData;                                    |
|      CHAR       IniApplicationName[2 ];                          |
|      PSZ        ExSearchPath;                                    |
|      BOOL       CountOnlyVisibleWindows;                         |
|      USHORT      ApplicationID;                                  |
|      CHAR       ApplicationName[2 ];                             |
|      ULONG       EditWindowListStyle;                            |
|      } GLOBDATA;                                                 |
|                                                                  |
+-------------------------------------------------------------------
```

hAB The Application Anchor Block. To obtain the applications anchor block use the WinInitialize function. (example: hab = WinInitialize(NULL);)

eres Structure shared between applications and ERES.DLL. The pointer to this structure is passed into ERES.DLL where it is used internally. (See Appendix G for a description of this structure)

eresModule Module handle of ERES.DLL. Optained with the DosLoadModule function.

hmq The handle to the applications message queue.

hDCApp The handle to the applications window device context.

DefFrameProc Used internally.

ShrMsgBuff The pointer to the applications shared message buffer.

rc Used in positioning editwindows.

stagcount Stagger window count.

LastCmdLine The last command line used to open an edit window.

EPMClass Edit window class name. Used to identify edit windows.

AllowMultProcesses Flag to allow multiple processes

hwndHelpInstance Window handle of the a help instance of the information presentation facility

hmiHelpData Hook for the information presentation facility.

IniApplicationName Name of the ini file to save and retrieve internal data from.

ExSearchPath Ex file search path.

CountOnlyVisibleWindows When set this flag insures that ERESCountEwindows returns only the number of visible edit windows.

ApplicationID Unique window id specified for editwindows.

ApplicationName Name of application.

EditWindowListStyle Edit window list dialog style flag. The following flags can be used to position the edit window list dialog.

- EWindow_List_Window_Middle - display in the middle of the active window.
- EWindow_List_Window_Corner - display in the lower right corner of the active window.

Appendix F -

Description of the Eres Data Structure. (EPMRES)

```
Eres Data Structure defined in 'C'
typedef struct EPMRES_TYPE {
   HAB    hAB;
   HWND   hwndAppFrame;
   HWND   hwndAppClient;
   HWND   hwndActiveEdit;
   HWND   hwndActiveDlgBox;
   HWND   hwndSearchReplaceBox;
   CHAR   exfilename[MAXFILENAME];
   LISTDLG cmd;
   LISTDLG msgbox;
   LISTDLG open1;
   BOOL   skipopen1;
   CHAR   saveretpath[MAXFILENAME];
   PVOID  dlgin;
   PVOID  internal;
} EPMRES;
```

hAB The Application Anchor Block. To obtain the applications anchor block use the WinInitialize function. (example: hab = WinInitialize(NULL);)

hwndAppFrame The applications frame window handle.

hwndAppClient The applications client window handle.

hwndActiveEdit The window handle of the active edit window.

hwndActiveDlgBox The window handle of the active dialog box.

hwndSearchReplaceBox The window handle of the Search Replace dialog box.

exfilename String containing the name of the main ex file.

cmd Structure used to store entries in the command dialog box.

msgbox Structure used to store entries in the Message box dialog.

open1 Structure containing previous entries in the open dialog box.

skipopen1 When set skips the open1 dialog box and presents the list dialog box.

saveretpath The name of the file containing retrieve list data.

dlgin Pointer to dialog input information.

internal Used internally in ERES.DLL.

Appendix G -

Description the messages handled by ERESCommonWndProc()

The Following message are handled by ERESCommonWndProc():

- WM_COMMAND
- EPM_EDIT_NEWFILE
- EPM_EDIT_HELPNOTIFY
- EPM_EDIT_RETCODE
- EPM_EDIT_DESTROYNOTIFY
- EPM_POPCMDLINE
- EPM_POPMSGBOX
- EPM_POPOPENDLG
- EPM_POPCHANGEDLG
- EPM_POPCONFIGDLG
- EPM_POPHELPBROWSER
- EPM_POPHELPMGRPANEL
- EPM_OPEN_EDITWINDOW
- EPM_SAVE_LISTS
- EPM_RETRIEVE_LISTS
- EPM_EDIT_ACTIVEHWND

WM_COMMAND

Process a message from the system menu bar. Message parameter one is the id of the item selected.

EPM_EDIT_NEWFILE An existing edit window is requesting to open up a new edit window. ERESOpenEditWindow is called to handle it.

EPM_EDIT_HELPNOTIFY This message is in response to a WM_HELP message that was received by an edit window. It is handled by either creating a help manager panel, or create a file with the help brower file.

EPM_EDIT_RETCODE

Used internally.

EPM_EDIT_DESTROYNOTIFY This message notifies EPM that a edit window has been closed. If it was the last edit window and the search replace dialog exists, destroy that dialog.

EPM_POPCMDLINE This message is handled by poping the command line dialog box.

EPM_POPMSGBOX This message is handled by poping the message dialog box.

EPM_POPOPENDLG This message is handled by poping the open1 dialog box.

EPM_POPCHANGEDLG This message is handled by poping the find/change dialog box.

EPM_POPCONFIGDLG This message is handled by poping the configuration dialog box.

EPM_POPHELPBROWSER This message is handled by poping an edit window containing the help browser file.

EPM_POPHELPMGRPANEL This message pops a help manager panel.

EPM_OPEN_EDITWINDOW Used internally by EPM

EPM_SAVE_LISTS Calls ERESSaveLists()

EPM_RETRIEVELIST Calls ERESRetrieveList()

EPM_EDIT_ACTIVEHWND Used internally.