EmEditor Home - EmEditor Help

Plug-in Reference

You can develop plug-ins that extend EmEditor functions by using the C programming language.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference

Functions to Export

OnCommand(HWND hwnd)	The plug-in has been selected from a menu or a toolbar.
QueryStatus(HWND hwnd, LPBOOL pbChecked)	Queries the status of the plug-in, whether the command is enabled and whether the plug-in is a checked status.
OnEvents(HWND hwnd, UINT nEvent, LPARAM lParam)	When a status is changed, this function is called with the Events parameter.
GetMenuTextID()	Retrieves a resource ID for the plug-in menu item text.
GetStatusMessageID()	Retrieves a resource ID for the status bar text combined with the tool bar tool tip text with \n.
GetBitmapID()	Obtains a bitmap resource ID for the plug-in displayed on a toolbar.
PlugInProc(HWND hwnd, UINT nMsg, WPARAM wParam, LPARAM lParam)	Uses Messages to Plug-ins to retrieve or set settings.

These export functions must be defined in a DEF File by this order. When you compile them, you need to select _stdcall as a calling method. For details see Samples.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference

Messages

EE_ADD_REF	Increments the reference number of the plug-in.
EE_CONVERT	Converts characters.
EE_CUSTOM_BAR_OPEN	Opens a custom bar.
EE_CUSTOM_BAR_CLOSE	Closes a custom bar.
EE_DEV_TO_VIEW	Converts the device (client) coordinates of a specified position to the display coordinates.
EE_DO_IDLE	Refreshes the toolbar, the window title, the tab, and others.
EE_EMPTY_UNDO_BUFFER	Empties the buffer for the Undo and Redo commands.
EE_ENUM_CONFIG	Enumerates available configurations.
EE_ENUM_HIGHLIGHT	Enumerates highlighted strings.
EE_EXEC_COMMAND	Executes a specified command.
EE_FIND_IN_FILESA	Searches for an ANSI string from multiple files in the specified path.
EE_FIND_IN_FILESW	Searches for a Unicode string from multiple files in the specified path.
EE_FINDA	Searches for an ANSI string.

EE_FINDW	Searches for an Unicode string.
EE_FIND_REGEX	Searches a string for a regular expression.
EE_FREE	Frees a specified plug-in.
EE_GET_ACCEL_ARRAY	Retrieves the array of the shortcut keys.
EE_GET_ANCHOR_POS	Retrieves the origin point of the selection.
EE_GET_CARET_POS	Retrieves the current cursor position.
EE_GET_CMD_ID	Obtains the Plug-in command ID.
EE_GET_CONFIGA	Retrieves the selected configuration name by an ANSI string.
EE_GET_CONFIGW	Retrieves the selected configuration name by a Unicode string.
EE_GET_LINEA	Retrieves the ANSI text on the specified line.
EE_GET_LINES	Retrieves the number of the lines in EmEditor.
EE_GET_LINEW	Retrieves the Unicode text on the specified line.
EE_GET_MARGIN	Retrieves the margin size.
EE_GET_MODIFIED	Retrieves the modified state of the text.
EE_GET_OUTLINE_LEVEL	Retrieves the outline level for the specified logical line.
EE_GET_PAGE_SIZE	Retrieves a page size.
EE_GET_REDRAW	Retrieves the flag that allows changes in EmEditor to be redrawn or prevents changes in EmEditor to be redrawn.
EE_GET_REF	Retrieves the reference number of a specified plug-in.
EE_GET_SCROLL_POS	Retrieves the current positions of the scroll bars.
EE_GET_SEL_END	Retrieves the ending character position of the selection.
EE_GET_SEL_START	Retrieves the starting character position of the selection.
EE_GET_SEL_TEXTA	Retrieves the selected ANSI text.
EE_GET_SEL_TEXTW	Retrieves the selected Unicode text.
EE_GET_SEL_TYPE	Obtains the type of selection status.
EE_GET_REDRAW	Retrieves the flag that allows changes in EmEditor to be redrawn or prevents changes in EmEditor to be redrawn.
EE_GET_STATUSA	Retrieves the ANSI text displayed on the status bar.
EE_GET_STATUSW	Retrieves the Unicode text displayed on the status bar.
EE_GET_VERSION	Returns the version number.
EE_HELP	Displays the specified page of the Help.
EE_INFO	Retrieves or sets the value of one of the information parameters used by EmEditor.
EE_INSERT_FILEA	Inserts the specified file contents at the cursor (ANSI).
EE_INSERT_FILEW	Inserts the specified file contents at the cursor (Unicode).
EE_INSERT_STRINGA	Inserts an ANSI string into the current cursor position.
EE_INSERT_STRINGW	Inserts a Unicode string into the current cursor position.
EE_IS_CHAR_HALF_OR_FULL	Queries whether a specified character is a half-width or full-width character.
EE_KEYBOARD_PROP	Displays the Keyboard Properties for the specified command ID and configuration.
EE_LINE_FROM_CHAR	Retrieves the index of the line that contains the specified character index (the serial position)
EE_LINE_INDEX	Retrieves the character index of the first character of a specified line in EmEditor
EE_LOAD_CONFIGA	Reloads a configuration of which name is specified by an ANSI string
EE_LOAD_CONFIGW	Reloads a configuration of which name is specified by a Unicode string
EE_LOAD_FILEA	Loads a specified file into EmEditor (ANSI)
EE_LOAD_FILEW	Loads a specified file into EmEditor (Unicode)
EE_LOGICAL_TO_SERIAL	converts the logical coordinates to the serial position
EE_LOGICAL_TO_VIEW	converts the logical coordinates to the display coordinates
EE_MATCH_REGEX	Searches a string for a regular expression.
EE_OUTPUT_DIR	Sets the current directory for the output bar.
EE_OUTPUT_STRING	Appends a string to the output bar.
EE_QUERY_STATUS	Queries the status of the command, whether the command is enabled and whether the command is a checked status.
EE_REDRAW	Allows changes in EmEditor to be redrawn or prevents changes in EmEditor to be redrawn.
EE_REG_SET_VALUE	Sets a value into the Registry or an INI file depending on the EmEditor settings.
EE_REG_QUERY_VALUE	Queries a value from the Registry or an INI file depending on the EmEditor settings.

EE_RELEASE	Decrements the reference number of the plug-in.
EE_REPLACE_IN_FILESA	Replaces an ANSI string in multiple files in the specified path.
EE_REPLACE_IN_FILESW	Replaces a Unicode string in multiple files in the specified path.
EE_REPLACEA	Replaces an ANSI string
EE_REPLACEW	Replaces an Unicode string
EE_SAVE_FILEA	Saves the text to a specified file(ANSI)
EE_SAVE_FILEW	Saves the text to a specified file(Unicode)
EE_SERIAL_TO_LOGICAL	converts the serial position to the logical coordinates
EE_SET_ANCHOR_POS	Sets the origin point of the selection.
EE_SET_CARET_POS	Moves the cursor position and optionally extends the selection.
EE_SET_CONFIGA	Changes to a configuration specified by an ANSI string
EE_SET_CONFIGW	Changes to a configuration specified by a Unicode string.
EE_SET_MODIFIED	Changes the modified state of the text.
EE_SET_SCROLL_POS	Specifies the scroll bars position.
EE_SET_SEL_LENGTH	Changes the character length of the selection.
EE_SET_SEL_TYPE	Sets the type of selection status.
EE_SET_SEL_VIEW	Changes the the starting and ending position of the selection.
EE_SET_OUTLINE_LEVEL	Sets the outline level for the specified logical line.
EE_SET_STATUSA	Displays an ANSI message on the status bar.
EE_SET_STATUSW	Displays a Unicode message on the status bar.
EE_SHOW_OUTLINE	Shows or hides the outline.
EE_TOOLBAR_CLOSE	Closes a custom toolbar.
EE_TOOLBAR_OPEN	Opens a custom toolbar.
EE_TOOLBAR_SHOW	Shows or hides a custom toolbar.
EE_UPDATE_TOOLBAR	Updates a button status in a toolbar.
EE_VIEW_TO_DEV	Converts the display coordinates of a specified position to the device (client) coordinates.
EE_VIEW_TO_LOGICAL	Converts the display coordinates of a specified position to the logical coordinates.

These constants are defined at the header file (plugin.h).

The following Windows API messages are also supported. Some of their supports may not be complete. As the window handle parameter for the message, specify the window handle of the view, not frame of EmEditor. See Microsoft MSDN library for more information on these messages.

EM_GETSEL	Retrieves the starting and ending character positions of the current selection.
EM_SCROLLCARET	Scrolls the cursor into view.
EM_SETSEL	Selects a range of characters.
EM_REPLACESEL	Replaces the current selection with the specified text.
WM_CLEAR	Deletes the current selection.
WM_COPY	Copies the current selection to the Clipboard.
WM_CUT	Deletes the current selection and copy the deleted text to the Clipboard.
WM_GETTEXT	Copies the entire document into a buffer.
WM_GETTEXTLENGTH	Retrieves the size of buffer needed to retrieve the entire document excluding a terminating null character.
WM_PASTE	Copies the current content of the clipboard at the current cursor position.
WM_SETTEXT	Sets the entire document.
WM_UNDO	Undo the last operation.

Send feedback on this topic to Emurasoft

EE_ADD_REF

Increments the reference number of the plug-in. You can send this message explicitly or use the Editor_AddRef inline function.

EE_ADD_REF
wParam = 0;

lParam = (LPARAM)(HINSTANCE)hInstance;

Parameters

hInstance

Specifies the instance handle for the plug-in.

Return Values

The return value is the reference number of the plug-in after incremented. If the return value is less than or equal to zero, the specified plug-in can be safely unloaded from EmEditor.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_CONVERT

Converts characters. You can send this message explicitly or use the Editor_Convert inline function.

EE_CONVERT
wParam = (WPARAM) (UINT) nFlags;
lParam = 0;

Parameters

nFlags

You can specify a combination of the following values.

Value	Meaning
FLAG_MAKE_LOWER	Converts to lowercase characters.
FLAG_MAKE_UPPER	Converts to uppercase characters.
FLAG_HAN_TO_ZEN	Converts to full-size characters.
FLAG_ZEN_TO_HAN.	Converts to half-size characters
FLAG_CAPITALIZE	Capitalizes the first letter of each word.
FLAG_CONVERT_SELECT_ALL	Converts the entire text. If this flag is not set, EE_CONVERT converts the characters only in the selection.
FLAG_CONVERT_KATA	Converts Katakana.
FLAG_CONVERT_ALPHANUMERIC	Converts Alphabets and numeric characters.
FLAG_CONVERT_MARK	Converts marks.
FLAG_CONVERT_SPACE	Converts spaces.
FLAG_CONVERT_KANA_MARK	Converts Kana marks.

Return Values

If the message succeeds, the return value is nonzero. If the message fails, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_CUSTOM_BAR_OPEN

Opens a custom bar. If a custom bar is already opened before sending this message, EmEditor closes the custom bar, and opens a new custom bar. You can send this message explicitly or use the Editor_CustomBarOpen inline function.

EE_CUSTOM_BAR_OPEN
wParam = 0;
lParam = (LPCTSTR) (LPCTSTR) pCustomBarInfo;

Parameters

pCustomBarInfo

Pointer to the CUSTOM_BAR_INFO structure.

Return Values

The return value is a custom bar ID, which is necessary when the custom bar is closed with the EE_CUSTOM_BAR_CLOSE message. If the message fails, the return value is zero.

Version

Supported on EmEditor Professional Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_CUSTOM_BAR_CLOSE

Closes a custom bar. You can send this message explicitly or use the Editor_CustomBarClose inline function.

```
EE_CUSTOM_BAR_CLOSE
   wParam = nCustomBarID;
   lParam = 0;
```

Parameters

nCustomBarID

Specifies the custom bar to close. This is the return value from the EE_CUSTOM_BAR_OPEN message.

Return Values

If the message succeeds, the return value is TRUE. If the message fails, the return value is FALSE.

Version

Supported on EmEditor Professional Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_DEV_TO_VIEW

Convert the device (client) coordinates of a specified position to the display coordinates. You can send this message explicitly or use the Editor_DevToView inline function.

```
EE_DEV_TO_VIEW
   wParam = (WPARAM) (POINT_PTR*) pptDev;
   lParam = (LPARAM) (POINT_PTR*) pptView;
```

Parameters

pptDev

Pointer to a POINT_PTR structure that specifies the device coordinates to be converted.

pptView

Pointer to a POINT_PTR structure to receive the converted display coordinates.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EE_DO_IDLE

Refreshes the toolbar, the window title, the tab, and others. You can send this message explicitly or use the Editor_DoIdle inline function.

```
EE_DO_IDLE
   wParam = (WPARAM) (BOOL) bResetTab;
   lParam = (LPARAM) 0;
```

Parameters

bResetTab

Resets the tab.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 6.00 or later.

Send feedback on this topic to Emurasoft

EE_EMPTY_UNDO_BUFFER

Empties the buffer for the Undo and Redo commands. You can send this message explicitly or use the Editor_EmptyUndoBuffer inline function.

```
EE_EMPTY_UNDO_BUFFER
   wParam = 0;
   lParam = 0;
```

Parameters

None.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EE_EXEC_COMMAND

Executes a specified command. You can send this message explicitly or use the Editor_ExecCommand inline function.

```
EE_EXEC_COMMAND
   wParam = (WPARAM) (UINT) nCmdID;
   lParam = 0;
```

Parameters

nCmdID

The identifier of the command to be executed. See Command IDs.

Return Value

The return value is not used.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_FIND_IN_FILESA

Searches for an ANSI string from multiple files in the specified path. The list of searched files will be displayed in the current window. If the current document is modified, displays the prompt message box whether to save the changes to the current file. You can send this message explicitly or use the Editor_FindInFilesA inline function.

```
EE_FIND_IN_FILESA
   wParam = 0;
   lParam = (LPARAM) (GREP_INFOA) pGrepInfo;
```

Parameters

pGrepInfo

Specifies a pointer to the GREP_INFOA Structure.

Return Value

Returns FALSE if the user aborts, or TRUE if not.

Version

Supported on EmEditor Professional Version 4.02 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_FIND_IN_FILESW

Searches for an Unicode string from multiple files in the specified path. The list of searched files will be displayed in the current window. If the current document is modified, displays the prompt message box whether to save the changes to the current file. You can send this message explicitly or use the Editor_FindInFilesW inline function.

```
EE_FIND_IN_FILESW
   wParam = 0;
   lParam = (LPARAM) (GREP_INFOW) pGrepInfo;
```

Parameters

pGrepInfo

Specifies a pointer to the GREP_INFOW Structure.

Return Value

Returns FALSE if the user aborts, or TRUE if not.

Version

Supported on EmEditor Professional Version 4.02 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_FIND_REGEX

Searches a string for a regular expression. You can send this message explicitly or use the Editor_FindRegex inline function.

EE_FIND_REGEX
wParam = 0;
lParam = (LPARAM) (FIND_REGEX_INFO*) pFindRegexInfo;

Parameters

pFindRegexInfo

Pointer to the FIND_REGEX_INFO structure.

Return Values

If a string that matches the specified regular expression is found, the return value is TRUE. If the specified regular expression is not found, the return value is FALSE. If the regular expression has an syntax error or another fatal error occurs, the return value is -1.

Version

Supported on Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_FINDA

Searches an ANSI string. You can send this message explicitly or use the Editor_FindA inline function.

EE_FINDA
wParam = (WPARAM) (UINT) nFlags;
lParam = (LPARAM) (LPCSTR) szFind;

Parameters

nFlags

You can specify a combination of the following values.

Value	Meaning
FLAG_FIND_NEXT	Searches the string downward from the cursor position. If this flag is not set, searches the string upward.
FLAG_FIND_CASE	Matches cases.
FLAG_FIND_ESCAPE	Uses escape sequences.
FLAG_FIND_NO_PROMPT	Suppresses displaying a dialog box even if no string is found.
FLAG_FIND_ONLY_WORD	Searches only words.
FLAG_FIND_AROUND	Moves to start/end of the text.
FLAG_FIND_REG_EXP	Uses a regular expression.
FLAG_FIND_CLOSE	Closes the dialog box after finished.

szFind

Specifies a string to search.

Return Values

If the message succeeds, the return value is nonzero. If the message fails, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_FINDW

Searches a Unicode string. You can send this message explicitly or by using the Editor_FindW inline function.

EE_FINDW
wParam = (WPARAM) (UINT) nFlags;
lParam = (LPARAM) (LPCWSTR) szFind;

Parameters

nFlags

You can specify a combination of the following values.

Value	Meaning
FLAG_FIND_NEXT	Searches the string downward from the cursor position. If this flag is not set, searches the string upward.
FLAG_FIND_CASE	Matches cases.
FLAG_FIND_ESCAPE	Uses escape sequences.
FLAG_FIND_NO_PROMPT	Suppresses displaying a dialog box even if no string is found.
FLAG_FIND_ONLY_WORD	Searches only words.
FLAG_FIND_AROUND	Moves to start/end of the text.
FLAG_FIND_REG_EXP	Uses a regular expression.
FLAG_FIND_CLOSE	Closes the dialog box after finished.

szFind

Specifies a string to search.

Return Values

If the message succeeds, the return value is nonzero. If the message fails, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_FREE

Frees a specified plug-in. You can send this message explicitly or by using the Editor_Free inline function.

EE_FREE
wParam = 0;
lParam = (LPARAM)(ATOM)atom;

Parameters

atom

Specifies the atom of a specified plug-in file name.

Return Values

If the plug-in is freed, the return value is TRUE. If the plug-in is not freed, the return value is FALSE.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_ACCEL_ARRAY

Retrieves the array of the shortcut keys. You can send this message explicitly or by using the Editor_GetAccelArray inline function.

EE_GET_ACCEL_ARRAY
wParam = (UINT) nBufSize;
lParam = (ACCEL*) pAccel;

Parameters

nBufSize

Specifies the size of the buffer, in ACCEL, that will receive the shortcut key arrays.

pAccel

Specifies the pointer to the buffer that receives the array of the ACCEL structures.

Return Values

The return value is the size of the buffer, in ACCEL, that is needed to receive the shortcut key arrays.

Version

Supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_ANCHOR_POS

Retrieves the origin point of the selection. You can send this message explicitly or by using the Editor_GetAnchorPos inline function.

```
EE_GET_ANCHOR_POS
   wParam = (WPARAM) (int) nLogical;
   lParam = (LPARAM) (POINT_PTR*) pptPos;
```

Parameters

nLogical

Specifies one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

pptPos

Pointer to a POINT_PTR structure that will receive the origin point of the selection.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_CARET_POS

Retrieves the current cursor position. You can send this message explicitly or by using the Editor_GetCaretPos inline function.

```
EE_GET_CARET_POS
   wParam = (WPARAM) (int) nLogical;
   lParam = (LPARAM) (POINT_PTR*) pptPos;
```

Parameters

nLogical

Specifies one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

pptPos

Pointer to a POINT_PTR structure that will receive the current cursor position.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_CMD_ID

Obtains the Plug-in command ID. You can send this message explicitly or by using the Editor_GetCmdID inline function.

```
EE_GET_CMD_ID
   wParam = 0;
   lParam = (LPARAM) (HINSTANCE) hInstance
```

Parameters

hInstance

Specifies the Plug-in instance handle.

Return Values

Returns command ID to run this Plug-in.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_CONFIGA

Retrieves the selected configuration name by an ANSI string. You can send this message explicitly or use the Editor_GetConfigA inline function.

```
EE_GET_CONFIGA
   wParam = 0;
   lParam = (LPARAM) (LPSTR) szConfigName;
```

Parameters

szConfigName

Specifies a buffer that will receive the configuration name. The buffer size must be at least MAX_CONFIG_NAME bytes.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_CONFIGW

Retrieves the selected configuration name by a Unicode string. You can send this message explicitly or use the Editor_DocGetConfigW inline function or the Editor_GetConfigW inline function.

```
EE_GET_CONFIGW
   wParam = MAKEWPARAM(0, (iDoc)+1);
   lParam = (LPARAM) (LPWSTR) szConfigName;
```

Parameters

iDoc

Specifies the index of the target document. A one-based index should be specified at the higher word of wParam. If 0 is specified at the higher word of wParam, the currently active document will be targeted.

szConfigName

Specifies a buffer that will receive the configuration name. The buffer size must be at least MAX_CONFIG_NAME in words.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_LINEA

Retrieves the ANSI text on the specified line. You can send this message explicitly or use the Editor_GetLineA inline function.

```
EE_GET_LINEA
   wParam = (WPARAM) (GET_LINE_INFO*) pGetLineInfo;
   lParam = (LPARAM) (LPSTR) szString;
```

Parameters

pGetLineInfo

Specifies the pointer to GET_LINE_INFO structure.

szString

Pointer to the buffer that will receive the text.

Return Values

If *pGetLineInfo->cch* is zero, the return value is the required size, in bytes, for a buffer that can receive the text. If *pGetLineInfo->cch*. is not zero, the return value is not used. If nBufferSize. is not zero, the return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_LINES

Retrieves the number of the lines for the specified document. You can send this message explicitly or use the Editor_DocGetLines inline function or the Editor_GetLines inline function.

```
EE_QUERY_STATUS
   wParam = (WPARAM) MAKEWPARAM(nLogical, iDoc+1);
   lParam = 0;
```

Parameters

nLogical

Specifies one of the following Values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

iDoc

Specifies the index of the target document. A one-based index should be specified at the higher word of wParam. If 0 is specified at the higher word of wParam, the currently active document will be targeted.

Return Values

Returns the number of the lines in EmEditor. If the last line is ended with a return, the last line will be counted. If the text is empty, returns one.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_LINEW

Retrieves the Unicode text on the specified line. You can send this message explicitly or use the Editor_GetLineW inline function.

```
EE_GET_LINEW
   wParam = (WPARAM) (GET_LINE_INFO*) pGetLineInfo;
   lParam = (LPARAM) (LPWSTR) szString;
```

Parameters

pGetLineInfo

Pointer to the GET_LINE_INFO structure.

szString

Pointer to the buffer that will receive the text.

Return Values

If *pGetLineInfo->cch* is zero, the return value is the required size, in words, for a buffer that can receive the text. If *pGetLineInfo->cch* is not zero, the return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_MARGIN

Retrieves the margin size. You can send this message explicitly or by using the Editor_GetMargin inline function.

```
EE_GET_MARGIN
   wParam = 0;
   lParam = 0;
```

Parameters

None.

Return Values

Returns the currently selected margin size. If the normal line margin size and the quoted line margin size are different, the larger margin will be returned. If lines are wrapped by the window size, the return value will depend on the current window size.

Send feedback on this topic to Emurasoft

EE_GET_MODIFIED

Retrieves the modified state of the text. You can send this message explicitly or use the Editor_GetModified inline function.

```
EE_GET_MODIFIED
  wParam = 0;
  lParam = 0;
```

Parameters

None.

Return Values

If the text is modified, the return value is TRUE. If the text is not modified, the return value is FALSE.

EE_GET_OUTLINE_LEVEL

Retrieves the outline level for the specified logical line. You can send this message explicitly or use the Editor_GetOutlineLevel inline function.

```
EE_GET_OUTLINE_LEVEL
  wParam = (WPARAM) (INT_PTR) nLogicalLine;
  lParam = 0;
```

Parameters

nLogicalLine

Specifies a logical line.

Return Values

The return value is the outline level for the specified logical line.

Version

Supported on EmEditor Professional Version 6.00 or later.

EE_GET_PAGE_SIZE

Retrieves a page size. You can send this message explicitly or use the Editor_GetPageSize inline function.

```
EE_GET_PAGE_SIZE
  wParam = 0;
  lParam = (LPARAM) (SIZE_PTR*) psizePage;
```

Parameters

psizePage

Pointer to a SIZE_PTR structure that will receive a page size. The page size is a pair of a number of lines and a number of columns that can display a page in the current EmEditor Window size.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EE_GET_REDRAW

Retrieves the flag that allows changes in EmEditor to be redrawn or prevents changes in EmEditor to be redrawn. You can send this message explicitly or use the Editor_GetRedraw inline function.

```
EE_GET_REDRAW
  wParam = (WPARAM) 0;
  lParam = (LPARAM) 0;
```

Parameters

None.

Return Values

Returns TRUE if the current flag allows changes in EmEditor to be redrawn or prevents changes in EmEditor to be redrawn. Otherwise, returns FALSE.

Version

Supported on Version 5.00 or later.

Send feedback on this topic to Emurasoft

EE_GET_REF

Retrieves the reference number of a specified plug-in. You can send this message explicitly or use the Editor_GetRef inline function.

```
EE_GET_REF
  wParam = 0;
  lParam = (LPARAM)(ATOM)atom;
```

Parameters

atom

Specifies the atom of a specified plug-in file name.

Return Values

The return value is the reference number of a specified plug-in. If the return value is zero, the specified plug-in can be safely unloaded from EmEditor.

Send feedback on this topic to Emurasoft

EE_GET_SCROLL_POS

Retrieves the current positions of the scroll bars. You can send this message explicitly or use the Editor_GetScrollPos inline function.

```
EE_GET_SCROLL_POS
  wParam = 0;
  lParam = (LPARAM) (POINT_PTR*) pptPos;
```

Parameters

pptPos

Pointer to a POINT_PTR structure that will receive the scroll bar positions.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EE_GET_SEL_END

Retrieves the ending character position of the selection. You can send this message explicitly or use the Editor_GetSelEnd inline function.

EE_GET_SEL_END
wParam = (WPARAM) (int) nLogical;
lParam = (LPARAM) (POINT_PTR*) pptPos;

Parameters

nLogical

Specifies one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

pptPos

Pointer to a POINT_PTR structure that will receive the ending character position of the selection.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EE_GET_SEL_START

Retrieves the starting character position of the selection. You can send this message explicitly or use the Editor_GetSelStart inline function.

EE_GET_SEL_START
wParam = (WPARAM) (int) nLogical;
lParam = (LPARAM) (POINT_PTR*) pptPos;

Parameters

nLogical

Specifies one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

pptPos

Pointer to a POINT_PTR structure that will receive the starting character position of the selection.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_SEL_TEXTA

Retrieves the selected ANSI text. You can send this message explicitly or by using the Editor_GetSelTextA inline function.

```
EE_GET_SEL_TEXTA
   wParam = (WPARAM) (UINT) nBufferSize;
   lParam = (LPARAM) (LPSTR) szBuffer;
```

Parameters

nBufferSize

Specifies the maximum number of characters in bytes to copy to the buffer, including the NULL character.

szBuffer

Pointer to the buffer that will receive the text.

Return Values

If *nBufferSize*. is zero, the return value is the required size, in bytes, for a buffer that can receive the text. If *nBufferSize*. is not zero, the return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_SEL_TEXTW

Retrieves the selected Unicode text. You can send this message explicitly or by using the Editor_GetSelTextW inline function.

```
EE_GET_SEL_TEXTW
   wParam = (WPARAM) (UINT) nBufferSize;
   lParam = (LPARAM) (LPWSTR) szBuffer;
```

Parameters

nBufferSize

Specifies the maximum number of characters in words to copy to the buffer, including the NULL character.

szBuffer

Pointer to the buffer that will receive the text.

Return Values

If *nBufferSize*. is zero, the return value is the required size, in words, for a buffer that can receive the text. If *nBufferSize* is not zero, the return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_SEL_TYPE

Obtains the type of selection status. You can send this message explicitly or by using the Editor_GetSelType inline function or Editor_GetSelTypeEx inline function.

EE_GET_SEL_TYPE
wParam = (WPARAM) (BOOL) bNeedAlways;
lParam = (LPARAM)0;

Parameters

bNeedAlways

If this parameter is TRUE, EE_GET_SEL_TYPE returns the type of selection status even if none is selected. If this parameter is FALSE, EE_GET_SEL_TYPE returns SEL_TYPE_NONE if none is selected.

Return Values

Returns a combination of the following values. SEL_TYPE_NONE, SEL_TYPE_CHAR, SEL_TYPE_LINE, and SEL_TYPE_BOX cannot be combined. SEL_TYPE_KEYBOARD and SEL_TYPE_SELECTED can be combined with other values. If bNeedAlways is TRUE and if text is selected, a logical sum with SEL_TYPE_SELECTED will be returned. If bNeedAlways is FALSE, SEL_TYPE_SELECTED will not be used.

Value	Meaning
SEL_TYPE_NONE	None is selected.
SEL_TYPE_CHAR	Characters are selected.
SEL_TYPE_LINE	Lines are selected.
SEL_TYPE_BOX	Boxes are selected.
SEL_TYPE_KEYBOARD	Selected by the keyboard.
SEL_TYPE_SELECTED	Selected (when bNeedAlways = TRUE)

Version

Supported on EmEditor Professional Version 3.00 or later. However, bNeedAlways is supported on Version 5.00 or later. On the previous versions, bNeedAlways is assumed to be FALSE.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_STATUSA

Retrieves the ANSI text displayed on the status bar. You can send this message explicitly or use the Editor_GetStatusA inline function.

EE_GET_STATUSA
wParam = nBufLen;
lParam = (LPARAM) (LPSTR) szMessage;

Parameters

nBufLen

Specifies the size of buffer in characters to retrieve the string including the terminating null character. You can specify 0 if *szMessage* is NULL. If the buffer size is not enough, *szMessage* will retrieve no string.

szMessage

Specifies the buffer to retrieve the string. If NULL is specified, returns the size of the buffer enough to retrieve the string.

Return Values

Returns TRUE if the current flag allows changes in EmEditor to be redrawn or prevents changes in EmEditor to be redrawn. Otherwise, returns FALSE.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_STATUSW

Retrieves the Unicode text displayed on the status bar. You can send this message explicitly or use the Editor_GetStatusW inline function.

EE_GET_STATUSW
wParam = nBufSize;
lParam = (LPARAM) (LPWSTR) szStatus;

Parameters

nBufSize

Specifies the size of buffer in characters to retrieve the string including the terminating null character. You can specify 0 if *szStatus* is NULL. If the buffer size is not enough, *szStatus* will retrieve no string.

szStatus

Specifies the buffer to retrieve the string. If NULL is specified, returns the size of the buffer enough to retrieve the string.

Return Values

Returns the size of the buffer in characters enough to retrieve the string including the terminating null character.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_GET_VERSION

Returns the version number. You can send this message explicitly or by using the Editor_GetVersion inline function.

EE_GET_VERSION
wParam = pnProductType;
lParam = 0;

Parameters

pnProductType

Specifies a pointer to an integer value. This message returns one of the following values.

VERSION_PRO	EmEditor Professional
VERSION_STD	EmEditor Standard

Return Values

Returns the version number multiplied by 1000.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_HELP

Displays the specified page of the Help. You can send this message explicitly or by using the Editor_Help inline function.

EE_HELP
wParam = 0
(LPCTSTR)lParam = szPageURL

Parameters

szPageURL

Specifies the URL of the Help page to display.

Return Values

Return value is not used.

Version

Supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_INFO

Retrieves or sets the value of one of the information parameters used by EmEditor. You can send this message explicitly or use the Editor_Info inline function or Editor_DocInfo inline function.

```
EE_INFO
   wParam = (WPARAM)(int)nCmd;
   lParam = (LPARAM)lParam;
```

or

```
EE_INFO
   wParam = MAKEWPARAM(nCmd, iDoc+1);
   lParam = (LPARAM)lParam;
```

Parameters

nCmd

Specifies a parameter to retrieve or set. This parameter can be one of the values from the following table.

nCmd	Meaning	lParam	Return Value
EI_GET_ENCODE	Retrieves the encoding method to save files.	Not used.	(int)nCP The encoding method.
EI_SET_ENCODE	Sets an encoding method to save files.	(UINT)nCP Specifies an encoding method, whose value begins by CODEPAGE_.	Not used.
EI_GET_SIGNATURE	Retrieves whether to sign Unicode/UTF-8 files.	Not used.	(BOOL)bSignature TRUE to sign.
EI_SET_SIGNATURE	Sets whether to sign Unicode/UTF-8 files.	(BOOL)bSignature TRUE to sign.	Not used.
EI_GET_FONT_CHARSET	Retrieves the character set to display.	Not used.	(int)nCharset The character set.
EI_SET_FONT_CHARSET	Sets a character set to display.	(int)nCharset Specifies an character set whose value begins by CHARSET_.	Not used.
EI_GET_FONT_CP	Retrieves the code tab used by the font to display.	Not used.	(UINT)nCP The code tab.
EI_GET_INPUT_CP	Retrieves the code tab used by the input languages.	Not used.	(UINT)nCP The code tab.
EI_GET_SHOW_TAG	Retrieves whether to show the tag highlighted.	Not used.	(BOOL)bShowTag TRUE to highlight the tag.
EI_SET_SHOW_TAG	Sets whether to show the tag highlighted.	(BOOL)bShowTag TRUE to highlight the tag.	Not used.
		(LPSTR)szFileName	

EI_GET_FILE_NAMEA	Retrieves the file name currently opened, in bytes.	Specifies a pointer to a buffer to retrieve the file name. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_FILE_NAMEW	Retrieves the file name currently opened, in Unicode.	(LPSTR)szFileName Specifies a pointer to a buffer to retrieve the file name. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_IS_PROPORTIONAL_FONT	Retrieves whether the display font is proportional.	Not Used.	(BOOL)bProportionalFont
EI_GET_NEXT_BOOKMARK	Finds the next book mark position.	(int)yLine Specifies an initial logical line to search from. -1 will search from the beginning of the document.	(int)yLine Returns the searched logical line. -1 will be returned if not found.
EI_GET_HILITE_FIND	Retrieves whether searched strings are highlighted.	Not used.	(BOOL)bShowFindHilite
EI_SET_HILITE_FIND	Sets whether searched strings are highlighted.	(BOOL)bShowFindHilite	Not used.
EI_GET_APP_VERSIONA	Retrieves the version name as an ANSI string.	(LPSTR)szVersionName Specifies a pointer to a buffer to retrieve the version string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_APP_VERSIONW	Retrieves the version name as a Unicode string.	(LPWSTR)szVersionName Specifies a pointer to a buffer to retrieve the version string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_READ_ONLY	Retrieves whether the document is read-only.	Not used.	(BOOL)bReadOnly
EI_IS_WINDOW_COMBINED	Retrieves whether the windows are combined .	Not used.	(BOOL)bCombined
EI_WINDOW_COMBINE	Sets whether the windows are combined .	(BOOL)bCombined Combines the windows if TRUE, or separate the windows if FALSE.	Not used.
EI_IS_UNDO_COMBINED	Retrieves whether an inserted string can be undone at once .	Not used.	(BOOL)bCombined
EI_GET_DOC_COUNT	Retrieves the number of opened documents in the current frame window (EmEditor Professional 5.00 or later only).	Not used.	(int)nCount Returns the number of documents.
	Converts a document		

EI_INDEX_TO_DOC	index to a document handle (EmEditor Professional 5.00 or later only).	Specifies the zero-based index of the document.	(HEEDOC)hDoc Returns the handle to the document.
EI_DOC_TO_INDEX	Converts a document handle to a document index.	Specifies the handle to the document.	(int)nIndex Returns the zero-based index of the document.
EI_ZORDER_TO_DOC	Converts a document z-order to a document handle.	Specifies the zero-based z-order of the document.	(HEEDOC)hDoc Returns the handle to the document.
EI_DOC_TO_ZORDER	Converts a document handle to a document z-order.	Specifies the handle to the document.	(int)nZOrder Returns the zero-based z-order of the document.
EI_GET_ACTIVE_INDEX	Retrieves the index of the active document.	Not used.	(int)nIndex Returns the zero-based index of the document.
EI_SET_ACTIVE_INDEX	Activates a document.	Not used.	(BOOL)bSuccess Returns TRUE if succeeded, or FALSE if failed.
EI_GET_FULL_TITLEA	Retrieves the full title of the document in ANSI string.	(LPSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_FULL_TITLEW	Retrieves the full title of the document in Unicode string.	(LPWSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_SHORT_TITLEA	Retrieves the short title of the document in ANSI string.	(LPSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_SHORT_TITLEW	Retrieves the short title of the document in Unicode string.	(LPWSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_SAVE_AS_TITLEA	Retrieves the full title of the document except the asterisk (*) indicating modification in ANSI string.	(LPSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_SAVE_AS_TITLEW	Retrieves the full title of the document except the asterisk (*) indicating modification in	(LPWSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the	Not used.

	Unicode string.	terminating NULL character.	
EI_MOVE_ORDER	Moves the document tab order	Specifies the zero-based index of the destination tab.	Not used.
EI_CLOSE_DOC	Closes the document.	Not used.	(BOOL)bSuccess Returns TRUE if succeeded, or FALSE if failed.
EI_SAVE_DOC	Saves the document. If the document is untitled, the **Save As** dialog box will appear.	Not used.	(BOOL)bSuccess Returns TRUE if succeeded, or FALSE if failed. Selecting Cancel in the **Save As** dialog box when the document is untitled will also return FALSE.
EI_GET_CURRENT_FOLDER	Retrieves the current working folder.	(LPWSTR)szDir Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_IS_LARGE_DOC	Retrieves the flag to indicate whether the document is very large.	Not used.	(BOOL)bLarge Returns TRUE if the document is very large. Otherwise, it returns FALSE.

iDoc

Specifies the index of the target document. A one-based index should be specified at the higher word of wParam. If 0 is specified at the higher word of wParam, the currently active document will be targeted. This parameter may not be used depending on nCmd. In this case, the higher word of wParam must be 0.

lParam

Depends on the parameter specified.

Return Values

Depends on the parameter specified.

Version

Supported on EmEditor Professional Version 3.00 or later. However, the iDoc parameter is supported on Version 5.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_INSERT_FILEA

Inserts the specified file contents at the cursor. The file name is specified as an ANSI string. You can send this message explicitly or use the Editor_InsertFileA inline function.

```
EE_INSERT_FILEA
   wParam = (WPARAM) (LOAD_FILE_INFO*) pLoadFileInfo;
   lParam = (LPARAM) (LPCSTR) szFileName;
```

Parameters

pLoadFileInfo

Pointer to a LOAD_FILE_INFO structure. If this parameter is NULL, EE_INSERT_FILEA will open a file by a method

predefined by the properties.

szFileName

Specifies a full path file name. If a non-existing file is specified, EE_INSERT_FILEA will fail.

Return Values

If the command is successful, the return value is nonzero. If the command it not successful, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_INSERT_FILEW

Inserts the specified file contents at the cursor. The file name is specified as a Unicode string. You can send this message explicitly or use the Editor_InsertFileW inline function.

EE_INSERT_FILEW
wParam = (WPARAM) (LOAD_FILE_INFO*) pLoadFileInfo;
lParam = (LPARAM) (LPCWSTR) szFileName;

Parameters

pLoadFileInfo

Pointer to a LOAD_FILE_INFO structure. If this parameter is NULL, EE_INSERT_FILEW will open a file by a method predefined by the properties.

szFileName

Specifies a full path file name. If a non-existing file is specified, EE_INSERT_FILEW will fail.

Return Values

If the command is successful, the return value is nonzero. If the command it not successful, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_INSERT_STRINGA

Inserts an ANSI string into the current cursor position. You can send this message explicitly or use the Editor_InsertStringA macro, Editor_InsertA macro, or Editor_OverwriteA inline function.

EE_INSERT_STRINGA
wParam = nInsertType;
lParam = (LPARAM) (LPCSTR) szString;

Parameters

nInsertType

Specifies a combination of the following values.

OVERWRITE_PER_PROP	Inserts or Overwrites depending on the current Insert/Overwrite status.
OVERWRITE_INSERT	Always inserts, and does not overwrite the existing string.
OVERWRITE_OVERWRITE	Always overwrites the existing string.
KEEP_SOURCE_RETURN_TYPE	Keep the return type (CR only, LF only or both CR and LF) specified in the szString parameter.
KEEP_DEST_RETURN_TYPE	Keep the destination return type (CR only, LF only or both CR and LF).

szString

Specifies the string to be inserted.

Return Values

The return value is not used.

Version

KEEP_SOURCE_RETURN_TYPE and KEEP_DEST_RETURN_TYPE flags are supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_INSERT_STRINGW

Inserts a Unicode string into the current cursor position. You can send this message explicitly or use the Editor_InsertStringW inline function, Editor_InsertW inline function, or Editor_OverwriteW inline function.

```
EE_INSERT_STRINGW
   wParam = nInsertType;
   lParam = (LPARAM) (LPCWSTR) szString;
```

Parameters

nInsertType

Specifies a combination of the following values.

OVERWRITE_PER_PROP	Inserts or Overwrites depending on the current Insert/Overwrite status.
OVERWRITE_INSERT	Always inserts, and does not overwrite the existing string.
OVERWRITE_OVERWRITE	Always overwrites the existing string.
KEEP_SOURCE_RETURN_TYPE	Keep the return type (CR only, LF only or both CR and LF) specified in the szString parameter.
KEEP_DEST_RETURN_TYPE	Keep the destination return type (CR only, LF only or both CR and LF).

szString

Specifies the string to be inserted.

Return Values

The return value is not used.

Version

KEEP_SOURCE_RETURN_TYPE and KEEP_DEST_RETURN_TYPE flags are supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_IS_CHAR_HALF_OR_FULL

Queries whether a specified character is a half-width or full-width character. You can send this message explicitly or use the Editor_IsCharHalfOrFull inline function.

```
EE_IS_CHAR_HALF_OR_FULL
   wParam = (WPARAM)(WCHAR)ch;
   lParam = (LPARAM)0;
```

Parameters

ch

Specifies the character code by Unicode to be queried.

Return Values

If the character is a full-width character, the return value is two. If the character is a half-width character, the return value is one.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_KEYBOARD_PROP

Displays the Keyboard Properties for the specified command ID and configuration. You can send this message explicitly or use the Editor_KeyboardProp inline function.

```
EE_KEYBOARD_PROP
   wParam = (WPARAM)(UINT)nCmdID;
   lParam = (LPARAM)(LPCWSTR)pszConfigName;
```

Parameters

nCmdID

Specifies the command ID for the initial selection on the Keyboard Properties.

pszConfigName

Specifies the configuration for which EmEditor displays the Keyboard Properties.

Return Values

If the user selects OK on the Configuration Properties, the return value is TRUE. If the user selects Cancel, the return value is FALSE.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_LINE_EROM_CHAR

Retrieves the index of the line that contains the specified character index (the serial position). A character index is the zero-based index of the character from the beginning of the entire text. You can send this message explicitly or use the Editor_LineFromChar inline function.

```
EE_LINE_FROM_CHAR
   wParam = (WPARAM) (int) nLogical;
   lParam = (LPARAM) (UINT) nSerialIndex;
```

Parameters

nLogical

Specify one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

nSerialIndex

Specifies the character index of the character contained in the line whose number is to be retrieved. If this parameter is -1, EE_LINE_FROM_CHAR retrieves the line number of the current line (the line containing the cursor).

Return Values

The return value is the zero-based line number of the line containing the character index specified by *nSerialIndex*.

Send feedback on this topic to Emurasoft

EE_LINE_INDEX

Retrieves the character index of the first character of a specified line in EmEditor. A character index is the zero-based index of the character from the beginning of the edit control. You can send this message explicitly or by using the Editor_LineIndex inline function.

```
EE_LINE_INDEX
   wParam = (WPARAM) (BOOL) bLogical;
   lParam = (LPARAM) (int) yLine;
```

Parameters

bLogical

Specifies TRUE if the line number is by the logical coordinates. Specifies FALSE if the line number is by the display coordinates.

yLine

Specifies the zero-based line number. A value of -1 specifies the current line number (the line that contains the cursor).

Return Values

The return value is the character index of the line specified in the *yLine* parameter.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_LOAD_CONFIGA

Reloads a configuration of which name is specified by an ANSI string. You can send this message explicitly or use the Editor_LoadConfigA inline function.

```
EE_LOAD_CONFIGA
   wParam = 0;
   lParam = (LPARAM) (LPCSTR) szConfigName;
```

Parameters

szConfigName

Specifies the name of a configuration to be reloaded.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_LOAD_CONFIGW

Reloads a configuration of which name is specified by a Unicode string. You can send this message explicitly or use the Editor_LoadConfigW inline function.

```
EE_LOAD_CONFIGW
   wParam = 0;
   lParam = (LPARAM) (LPCWSTR) szConfigName;
```

Parameters

szConfigName

Specifies the name of a configuration to be reloaded.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_LOAD_FILEA

Loads a specified file into EmEditor. The file name is specified by an ANSI string. You can send this message explicitly or use the Editor_LoadFileA inline function.

```
EE_LOAD_FILEA
   wParam = (WPARAM) (LOAD_FILE_INFO*) pLoadFileInfo;
   lParam = (LPARAM) (LPCSTR) szFileName;
```

Parameters

pLoadFileInfo

Pointer to a LOAD_FILE_INFO structure. If this parameter is NULL, EE_LOAD_FILEA will open a file by a method predefined by the properties.

szFileName

Specifies a full path file name in bytes. If a non-existing file is specified, EE_LOAD_FILEA will fail.

Return Values

If the command is enable, the return value is nonzero. If the command it not enable, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_LOAD_FILEW

Loads a specified file into EmEditor. The file name is specified by an ANSI string. You can send this message explicitly or use the Editor_LoadFileW inline function.

```
EE_LOAD_FILEW
   wParam = (WPARAM) (LOAD_FILE_INFO*) pLoadFileInfo;
   lParam = (LPARAM) (LPCWSTR) szFileName;
```

Parameters

pLoadFileInfo

Pointer to a LOAD_FILE_INFO structure. If this parameter is NULL, EE_LOAD_FILEW will open a file by a method predefined by the properties.

szFileName

Specifies a full path file name in bytes. If a non-existing file is specified, EE_LOAD_FILEW will fail.

Return Values

If the command is enable, the return value is nonzero. If the command it not enable, the return value is zero.

Send feedback on this topic to Emurasoft

EE_LOGICAL_TO_SERIAL

Convert the logical coordinates to the serial position. The serial position is the zero-based index of the character from the beginning of the entire text. You can send this message explicitly or use the Editor_LogicalToSerial inline function.

```
EE_LOGICAL_TO_SERIAL
   wParam = 0;
   lParam = (LPARAM) (POINT_PTR*) pptLogical
```

Parameters

pptLogical

Pointer to a POINT_PTR structure that specifies the logical coordinates to be converted.

Return Values

Return the serial position.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_LOGICAL_TO_VIEW

Convert the logical coordinates to the display coordinates. You can send this message explicitly or use the Editor_LogicalToView inline function.

```
EE_LOGICAL_TO_VIEW
   wParam = (WPARAM) (POINT_PTR*) pptLogical;
   lParam = (LPARAM) (POINT_PTR*) pptView;
```

Parameters

pptLogical

Pointer to a POINT_PTR structure that specifies the logical coordinates to be converted.

pptView

Pointer to a POINT_PTR structure to receive the converted display coordinates.

Return Values

The return value is not used.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_MATCH_REGEX

Determines whether a string matches a specified regular expression. You can send this message explicitly or use the Editor_MatchRegex inline function.

```
EE_MATCH_REGEX
   wParam = 0;
   lParam = (LPARAM) (MATCH_REGEX_INFO*) pMatchRegexInfo;
```

Parameters

pMatchRegexInfo

Pointer to the MATCH_REGEX_INFO structure.

Return Values

If a string matches the specified regular expression, the return value is TRUE. If a string does not match the specified regular expression, the return valud is FALSE. If the regular expression has an syntax error or another fatal error occurs, the return value is -1.

Version

Supported on Version 6.00 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_OUTPUT_DIR

Sets the current directory for the output bar. You can send this message explicitly or use the Editor_OutputDir inline function.

```
EE_OUTPUT_DIR
  wParam = 0;
  lParam = (LPARAM) (LPCWSTR) szCurrDir;
```

Parameters

szCurrDir

Specifies the current directory. This information is necessary if the text contains a clickable relative path that can be jumped only from the current directory.

Return Values

The return value is not used.

Version

Supported on EmEditor Version 7.00 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_OUTPUT_STRING

Appends a string to the output bar. You can send this message explicitly or use the Editor_OutputString inline function.

```
EE_OUTPUT_STRING
   wParam = nFlags;
   lParam = (LPARAM) (LPCWSTR) szString;
```

Parameters

nFlags

Specifies a combination of the following values.

FLAG_OPEN_OUTPUT	Opens the output bar.
FLAG_CLOSE_OUTPUT	Closes the output bar.
FLAG_FOCUS_OUTPUT	Sets the keyboard focus to the output bar.
FLAG_CLEAR_OUTPUT	Clears the contents of the output bar.

szString

Specifies the string to be appended.

Return Values

If the message succeeds, the return value is nonzero. If the message fails, the return value is zero.

Version

Supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_ENUM_CONFIG

Enumerates available configurations. You can send this message explicitly or use the Editor_EnumConfig inline function.

```
EE_ENUM_CONFIG
   wParam = (WPARAM) (size_t) cchBuf;
   lParam = (LPARAM) (LPWSTR) pBuf;
```

Parameters

cchBuf

Specifies the size of the buffer in characters. Note that two null characters will be added at the end of the list of configurations. If 0 is specified, this message returns the size necessary to retrieve the list of configurations.

pBuf

Specifies the pointer to the buffer to retrieve the list of configurations. In this buffer, the list of available configurations each separated by a null character will be retrieved. Two null characters will be added at the end of the list of configurations. If 0 is specified, pBuf can be NULL.

Return Values

The return value is the size necessary to retrieve the list of configurations.

Version

Supported on Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_ENUM_HIGHLIGHT

Enumerates highlighted strings. You can send this message explicitly or use the Editor_EnumHighlight inline function.

```
EE_ENUM_HIGHLIGHT
   wParam = (WPARAM) (size_t) cchBuf;
   lParam = (LPARAM) (LPWSTR) pBuf;
```

Parameters

cchBuf

Specifies the size of the buffer in characters. Note that two null characters will be added at the end of the list of highlighted strings. If 0 is specified, this message returns the size necessary to retrieve the list of highlighted strings.

pBuf

Specifies the pointer to the buffer to retrieve the list of highlighted strings. In this buffer, the list of highlighted strings each separated by a null character will be retrieved. Two null characters will be added at the end of the list of highlighted strings. If 0 is specified, pBuf can be NULL.

The first character of each string represents the color and a combination of the following values.

From 0 to 9	color. Use HIGHLIGHT_COLOR_MASK for mask.

HIGHLIGHT_WORD	whole world only.
HIGHLIGHT_RIGHT_SIDE	highlight right side.
HIGHLIGHT_INSIDE_TAG	inside tag only.
HIGHLIGHT_REG_EXP	regular expression.
HIGHLIGHT_CASE	match case.
HIGHLIGHT_RIGHT_ALL	highlight right all.

Return Values

The return value is the size necessary to retrieve the list of highlighted strings.

Version

Supported on Version 7.00 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_QUERY_STATUS

Queries the status of the command, whether the command is enable and whether the command is a checked status. You can send this message explicitly or by using the Editor_QueryStatus inline function.

EE_QUERY_STATUS
wParam = (WPARAM) (UINT) nCmdID;
lParam = (LPARAM) (BOOL*) pbChecked;

Parameters

nCmdID

The identifier of the command on which the status is queried. See Command IDs.

pbChecked

Pointer to a variable that receives a checked status (TRUE indicates the command is checked, FALSE indicates the command is not checked).

Return Values

If the command is enable, the return value is nonzero. If the command it not enable, the return value is zero.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_REDRAW

Allows changes in EmEditor to be redrawn or prevents changes in EmEditor to be redrawn. You can send this message explicitly or use the Editor_Redraw inline function.

EE_REDRAW
wParam = (WPARAM)bRedraw;
lParam = (LPARAM)0;

Parameters

bRedraw

Specifies the redraw state. If this parameter is TRUE, the content can be redrawn after a change. If this parameter is FALSE, the content cannot be redrawn after a change.

Return Values

The return value is not used.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_REG_SET_VALUE

Sets a value into the Registry or an INI file depending on the EmEditor settings. You can send this message explicitly or by using the Editor_RegSetValue inline function.

```
EE_REG_SET_VALUE
  wParam = 0;
  (REG_SET_VALUE_INFO*)lParam = pRegSetValueInfo;
```

Parameters

pRegSetValueInfo

Pointer to the REG_SET_VALUE_INFO structure.

Return Values

If the message succeeds, the return value is ERROR_SUCCESS.

If the message fails, the return value is a nonzero error code defined in Winerror.h.

Version

Supported on EmEditor Version 7.00 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_REG_QUERY_VALUE

Retrieves the data for the specified value from the Registry or an INI file depending on the EmEditor settings. You can send this message explicitly or by using the Editor_RegQueryValue inline function.

```
EE_REG_QUERY_VALUE
  wParam = 0;
  (REG_QUERY_VALUE_INFO*)lParam = pRegQueryValueInfo;
```

Parameters

pRegSetValueInfo

Pointer to the REG_QUERY_VALUE_INFO structure.

Return Values

If the message succeeds, the return value is ERROR_SUCCESS.

If the message fails, the return value is a nonzero error code defined in Winerror.h.

Version

Supported on EmEditor Version 7.00 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_RELEASE

Decrements the reference number of the plug-in. You can send this message explicitly or use the Editor_Release inline function.

EE_RELEASE
wParam = 0;
lParam = (LPARAM)(HINSTANCE)hInstance;

Parameters

hInstance

Specifies the instance handle for the plug-in.

Return Values

The return value is the reference number of the plug-in after decremented. If the return value is less than or equal to zero, the specified plug-in can be safely unloaded from EmEditor.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_REPLACE_IN_FILESA

Searches for an ANSI string from multiple files in the specified path. The list of searched files will be displayed in the current window. If the current document is modified, displays the prompt message box whether to save the changes to the current file. You can send this message explicitly or use the Editor_ReplaceInFilesA inline function.

EE_REPLACE_IN_FILESA
wParam = 0;
lParam = (LPARAM) (GREP_INFOA) pGrepInfo;

Parameters

pGrepInfo

Specifies a pointer to the GREP_INFOW Structure.

Return Value

Returns FALSE if the user aborts, or TRUE if not.

Version

Supported on EmEditor Professional Version 4.02 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_REPLACE_IN_FILESW

Replaces a Unicode string in multiple files in the specified path. You can send this message explicitly or use the Editor_ReplaceInFilesW inline function.

EE_REPLACE_IN_FILESW
wParam = 0;
lParam = (LPARAM) (GREP_INFOW) pGrepInfo;

Parameters

pGrepInfo

Specifies a pointer to the GREP_INFOW Structure.

Return Value

Returns FALSE if the user aborts, or TRUE if not.

Version

Supported on EmEditor Professional Version 4.02 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_REPLACEA

Replaces an ANSI string. You can send this message explicitly or use the Editor_ReplaceA inline function.

EE_REPLACEA
wParam = (WPARAM) (UINT) nFlags;
lParam = (LPARAM) (LPCSTR) szFindReplace;

Parameters

nFlags

You can specify a combination of the following values.

Value	Meaning
FLAG_FIND_CASE	Matches case.
FLAG_FIND_ESCAPE	Uses escape sequences.
FLAG_REPLACE_SEL_ONLY	Replaces only in the selection when specified with FLAG_REPLACE_ALL.
FLAG_REPLACE_ALL	Replaces all occurrences.
FLAG_FIND_NO_PROMPT	Suppresses displaying a dialog box even if no string is found.
FLAG_FIND_ONLY_WORD	Searches only words.
FLAG_FIND_REG_EXP	Uses a regular expression.
FLAG_FIND_CLOSE	Closes the dialog box after finished.

szFindReplace

Specifies a string to search and a string to replace to. You must specify the string to search and the string to replace to in this order, and the null character ('\0') must be inserted between the two.

Return Values

If the message succeeds, the return value is nonzero. If the message fails, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_REPLACEW

Replaces a Unicode string. You can send this message explicitly or by using the Editor_ReplaceW inline function.

EE_REPLACEW
wParam = (WPARAM) (UINT) nFlags;
lParam = (LPARAM) (LPCWSTR) szFindReplace;

Parameters

nFlags

You can specify a combination of the following values.

Value	Meaning
FLAG_FIND_CASE	Matches case.
FLAG_FIND_ESCAPE	Uses escape sequences.
FLAG_REPLACE_SEL_ONLY	Replaces only in the selection when specified with FLAG_REPLACE_ALL.
FLAG_REPLACE_ALL	Replaces all occurrences.
FLAG_FIND_NO_PROMPT	Suppresses displaying a dialog box even if no string is found.
FLAG_FIND_ONLY_WORD	Searches only words.
FLAG_FIND_REG_EXP	Uses a regular expression.
FLAG_FIND_CLOSE	Closes the dialog box after finished.

szFindReplace

Specifies a string to search and a string to replace to. You must specify the string to search and the string to replace to in this order, and the null character ('\0') must be inserted between the two.

Return Values

If the message succeeds, the return value is nonzero. If the message fails, the return value is zero.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_SAVE_FILEA

Saves the text to a specified file. The file name is specified by an ANSI string. You can send this message explicitly or use the Editor_SaveFileA inline function.

```
EE_SAVE_FILEA
   wParam = (WPARAM) (BOOL) bReplace;
   lParam = (LPARAM) (LPSTR) szFileName;
```

Parameters

bReplace

Specifies TRUE if the text will be saved by a specified name, and the file name EmEditor holds. The title shown on the EmEditor Window will be changed. Specifies FALSE if the copy of the text is saved, and the file name EmEditor holds will not be changed.

szFileName

Specifies a full path file name in bytes.

Return Values

If it is succeeded, the return value is nonzero. If it isn't succeeded, the return value is zero.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_SAVE_FILEW

Saves the text to a specified file. The file name is specified by an ANSI string. You can send this message explicitly or use the Editor_SaveFileW inline function.

```
EE_SAVE_FILEW
   wParam = (WPARAM) (BOOL) bReplace;
   lParam = (LPARAM) (LPWSTR) szFileName;
```

Parameters

bReplace

Specifies TRUE if the text will be saved as by a specified name, and the file name EmEditor holds and the title shown on the EmEditor Window will be changed. Specifies FALSE if the copy of the text is saved, and the file name EmEditor holds will not be changed.

szFileName

Specifies a full path file name in bytes.

Return Values

If it is succeeded, the return value is nonzero. If it isn't succeeded, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_SERIAL_TO_LOGICAL

Convert the serial position to the logical coordinates. The serial position is the zero-based index of the character from the beginning of the entire text. You can send this message explicitly or use the Editor_SerialToLogical inline function.

```
EE_SERIAL_TO_LOGICAL
   wParam = (WPARAM) (UINT) nSerial;
   lParam = (LPARAM) (POINT_PTR*) pptLogical;
```

Parameters

nSerial

Specifies a serial position to be converted.

pptLogical

Pointer to a POINT_PTR structure that will receive the converted logical coordinates.

Return Values

Return the serial potion.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_SET_ANCHOR_POS

Sets the origin point of the selection. You can send this message explicitly or by using the Editor_SetAnchorPos inline function.

```
EE_SET_ANCHOR_POS
   wParam = (WPARAM) (int) nLogical;
   lParam = (LPARAM) (POINT_PTR*) pptPos;
```

Parameters

nLogical

Specifies one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

pptPos

Pointer to a POINT_PTR structure that specifies the origin point of the selection.

Return Values

The return value is not used.

EE_SET_CARET_POS

Moves the cursor position and optionally extends the selection. You can send this message explicitly or by using the Editor_SetCaretPos inline function or the Editor_SetCaretPosEx inline function.

```
EE_SET_CARET_POS
   wParam = MAKEWPARAM( nLogical, bExtend );
   lParam = (LPARAM) (POINT_PTR*) pptPos;
```

Parameters

nLogical

Specifies a combination of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)
POS_SCROLL_DONT_CARE	The cursor position becomes where the scrolling becomes minimum.
POS_SCROLL_CENTER	The cursor position becomes near the center of the window.
POS_SCROLL_TOP	The cursor position becomes the top of the window.

bExtend

Determines whether to extend the current selection. If *bExtend* is TRUE, then the active end of the selection moves to the location while the anchor end remains where it is. Otherwise, both ends are moved to the specified location.

pptPos

Pointer to a POINT_PTR structure that specified the cursor position.

Return Values

The return value is not used.

Version

Supported on Version 4.03 or later. However, POS_SCROLL_DONT_CARE, POS_SCROLL_CENTER, and POS_SCROLL_TOP flags are supported on Version 6.00 or later.

EE_SET_CONFIGA

Changes to a configuration specified by an ANSI string. You can send this message explicitly or use the Editor_SetConfigA inline function.

```
EE_SET_CONFIGA
   wParam = 0;
   lParam = (LPARAM) (LPCSTR) szConfigName;
```

Parameters

szConfigName

Specifies a configuration by an ANSI string.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_SET_CONFIGW

Changes to a configuration specified by a Unicode string. You can send this message explicitly or use the Editor_SetConfigW inline function.

```
EE_SET_CONFIGW
   wParam = 0;
   lParam = (LPARAM) (LPCWSTR) szConfigName;
```

Parameters

szConfigName

Specifies a configuration by a Unicode string.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_SET_MODIFIED

Changes the modified state of the text. You can send this message explicitly or by using the Editor_SetModified inline function.

```
EE_SET_MODIFIED
   wParam = (WPARAM) (BOOL) bModified;
   lParam = 0;
```

Parameters

bModified

TRUE to change the state as modified, or FALSE to change the state as unmodified.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_SET_OUTLINE_LEVEL

Sets the outline level for the specified logical line. You can send this message explicitly or use the Editor_SetOutlineLevel inline function.

```
EE_SET_OUTLINE_LEVEL
   wParam = (WPARAM) (INT_PTR) nLogicalLine;
   lParam = (LPARAM) (int) nLevel;
```

Parameters

nLogicalLine

Specifies a logical line.

nLevel

Specifies an outline level.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 6.00 or later.

Send feedback on this topic to Emurasoft

EE_SET_SCROLL_POS

Specifies the scroll bars position. You can send this message explicitly or by using the Editor_SetScrollPos inline function or Editor_SetScrollPosEx inline function.

```
EE_SET_SCROLL_POS
   wParam = (WPARAM) (BOOL) bCanMoveCursor;
   lParam = (LPARAM) (POINT_PTR*) pptPos;
```

Parameters

bCanMoveCursor

If this parameter is TRUE and if the Move Cursor by Scrolling check box is selected, the cursor position will also move. If this parameter is FALSE, the cursor position will not move.

pptPos

Pointer to a POINT_PTR structure that specifies the scroll bar positions. The cursor position will not be changed.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 3.00 or later. However, bCanMoveCursor is supported on Version 5.00 or later. On the previous versions, bCanMoveCursor is assumed to be FALSE.

Send feedback on this topic to Emurasoft

EE_SET_SEL_LENGTH

Changes the character length of the selection. You can send this message explicitly or use the Editor_SetSelLength inline function.

```
EE_SET_SEL_LENGTH
   wParam = (WPARAM) (UINT) nLen;
   lParam = 0;
```

Parameters

nLen

Specifies the character length of the selection. Returns are always two character length (CR+LF).

Return Values

The return value is not used.

EE_SET_SEL_TYPE

Sets the type of selection status. You can send this message explicitly or use the Editor_SetSelType inline function or Editor_SetSelTypeEx inline function.

```
EE_SET_SEL_TYPE
   wParam = (WPARAM) (BOOL) bNeedAlways;
   lParam = (LPARAM) nSelType;
```

Parameters

bNeedAlways

If this parameter is TRUE, the type of selection status can be set even if none is selected. If this parameter is FALSE, SEL_TYPE_NONE will cancel the selection.

nSelType

You can specify a combination of the following values. However, SEL_TYPE_NONE, SEL_TYPE_CHAR, SEL_TYPE_LINE, SEL_TYPE_BOX cannot be combined. Only SEL_TYPE_KEYBOARD can be combined with SEL_TYPE_NONE, SEL_TYPE_CHAR, SEL_TYPE_LINE, or SEL_TYPE_BOX.

SEL_TYPE_NONE	Not selected.
SEL_TYPE_CHAR	Stream selection mode.
SEL_TYPE_LINE	Line selection mode.
SEL_TYPE_BOX	Box selection mode.
SEL_TYPE_KEYBOARD	Specifies the keyboard selection mode. This value can be combined with another value.

Return Values

Not used.

Version

Supported on EmEditor Professional Version 3.00 or later. However, bNeedAlways is supported on Version 5.00 or later. On the previous versions, bNeedAlways is assumed to be FALSE.

EE_SET_SEL_VIEW

Changes the the starting and ending position of the selection. You can send this message explicitly or use the Editor_SetSelView inline function.

```
EE_SET_SEL_VIEW
   wParam = (WPARAM) (POINT_PTR*) pptSelStart;
   lParam = (LPARAM) (POINT_PTR*) pptSelEnd;
```

Parameters

pptSelStart

Pointer to a POINT_PTR structure that specifies the starting position of the selection. The position is by display coordinates.

pptSelEnd

Pointer to a POINT_PTR structure that specifies the ending position of the selection. The position is by display coordinates.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_SET_STATUSA

Displays an ANSI message on the status bar. You can send this message explicitly or use the Editor_SetStatusA inline function.

```
EE_SET_STATUSA
   wParam = 0;
   lParam = (LPARAM) (LPCSTR) szStatus;
```

Parameters

szStatus

Specifies a massage text to be displayed on the status bar.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_SET_STATUSW

Displays a Unicode message on the status bar. You can send this message explicitly or use the Editor_SetStatusW inline function.

```
EE_SET_STATUSW
   wParam = 0;
   lParam = (LPARAM) (LPCWSTR) szStatus;
```

Parameters

szStatus

Specifies a massage text to be displayed on the status bar.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_SHOW_OUTLINE

Shows or hides the outline. You can send this message explicitly or use the Editor_ShowOutline inline function.

```
EE_SHOW_OUTLINE
   wParam = (WPARAM) (INT_PTR) nFlags;
   lParam = 0;
```

Parameters

nFlags

Specifies one of the following values.

Value	Meaning
SHOW_OUTLINE_SHOW	Shows outline.
SHOW_OUTLINE_HIDE	Hides outline.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_TOOLBAR_CLOSE

Closes a custom toolbar. You can send this message explicitly or by using the Editor_ToolbarClose inline function.

```
EE_TOOLBAR_CLOSE
  (UINT)wParam = nToolbarID
```

Parameters

nToolbarID

Specifies the toolbar to close. This is the return value from the EE_TOOLBAR_OPEN message.

Return Values

If the message succeeds and the toolbar state has been changed, the return value is TRUE. If the message fails or the toolbar state has not been changed, the return value is FALSE.

Version

Supported on EmEditor Professional Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_TOOLBAR_OPEN

Opens a custom toolbar. You can send this message explicitly or by using the Editor_ToolbarOpen inline function.

```
EE_TOOLBAR_OPEN
  wParam = 0;
  lParam = (LPARAM) (TOOLBAR_INFO*) pToolbarInfo;
```

Parameters

pToolbarInfo

Pointer to the TOOLBAR_INFO structure.

Return Values

The return value is a custom toolbar ID. If the message fails, the return value is zero.

Version

Supported on EmEditor Professional Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_TOOLBAR_SHOW

Shows or hides a custom toolbar. You can send this message explicitly or by using the Editor_ToolbarShow inline function.

```
EE_TOOLBAR_SHOW
   (UINT)wParam = nToolbarID
   (BOOL)lParam = bVisible
```

Parameters

nToolbarID

Specifies the toolbar to close. This is the return value from the EE_TOOLBAR_OPEN message.

bVisible

Specifies TRUE if the toolbar should be visible, or FALSE if the toolbar should be hidden.

Return Values

If the message succeeds and the toolbar state has been changed, the return value is TRUE. If the message fails or the toolbar state has not been changed, the return value is FALSE.

Version

Supported on EmEditor Professional Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_UPDATE_TOOLBAR

Updates a button status in a toolbar. You can send this message explicitly or by using the Editor_UpdateToolbar inline function.

```
EE_UPDATE_TOOLBAR
   wParam = (WPARAM) (UINT) nCmdID;
   lParam = 0;
```

Parameters

nCmdID

The identifier of the command on which the status is queried. See Command IDs.

Return Values

Return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_VIEW_TO_DEV

Converts the display coordinates of a specified position to the device (client) coordinates. You can send this message explicitly or use the Editor_ViewToDev inline function.

```
EE_VIEW_TO_DEV
```

wParam = (WPARAM) (POINT_PTR*) pptView;
lParam = (LPARAM) (POINT_PTR*) pptDev;

Parameters

pptView

Pointer to a POINT_PTR structure that specifies the display coordinates to be converted.

pptDev

Pointer to a POINT_PTR structure to receive the converted device coordinates.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages

EE_VIEW_TO_LOGICAL

Convert the display coordinates of a specified position to the logical coordinates. You can send this message explicitly or use the Editor_ViewToLogical inline function.

EE_VIEW_TO_LOGICAL
wParam = (WPARAM) (POINT_PTR*) pptView;
lParam = (LPARAM) (POINT_PTR*) pptLogical;

Parameters

pptView

Pointer to a POINT_PTR structure that specifies the display coordinates to be converted.

pptLogical

Pointer to a POINT_PTR structure that will receive the converted logical coordinates.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference

Inline Functions

Editor_AddRef	Increments the reference number of the plug-in.
Editor_Convert	Converts characters.
Editor_CustomBarOpen	Opens a custom bar.
Editor_CustomBarClose	Closes a custom bar.
Editor_DevToView	Converts the device (client) coordinates of a specified position to the display coordinates.
Editor_DocGetConfigA	Retrieves the selected configuration name for the specified document as an ANSI string.
Editor_DocGetConfigW	Retrieves the selected configuration name for the specified document as a Unicode string.
Editor_DocGetLines	Retrieves the number of the lines for the specified document.
Editor_DocGetModified	Retrieves the modified state of the text of the specified document.
Editor_DocInfo	Retrieves or sets the value of one of the information parameters used by EmEditor.
Editor_DocSaveFileA	Saves the text of the specified document to a specified file (ANSI).
Editor_DocSaveFileW	Saves the text of the specified document to a specified file (Unicode).
Editor_DocSetConfigA	Changes the specified document to a configuration specified by an ANSI string.

Editor_DocSetConfigW	Changes the specified document to a configuration specified by a Unicode string.
Editor_DoIdle	Refreshes the toolbar, the window title, the tab, and others.
Editor_EnumConfig	Enumerates available configurations.
Editor_EnumHighlight	Enumerates highlighted strings.
Editor_EmptyUndoBuffer	Empties the buffer for the Undo and Redo commands.
Editor_ExecCommand	Executes a specified command.
Editor_FindA	Searches an ANSI string.
Editor_FindInFilesA	Searches for an ANSI string from multiple files in the specified path.
Editor_FindInFilesW	Searches for a Unicode string from multiple files in the specified path.
Editor_FindRegex	Searches a string for a regular expression.
Editor_FindW	Searches an Unicode string.
Editor_Free	Frees a specified plug-in.
Editor_GetAccelArray	Retrieves the array of the shortcut keys.
Editor_GetAnchorPos	Retrieves the origin point of the selection.
Editor_GetCaretPos	Retrieves the current cursor position.
Editor_GetCmdID	Obtains the plug-in command ID.
Editor_GetConfigA	Retrieves the selected configuration name as an ANSI string.
Editor_GetConfigW	Retrieves the selected configuration name as a Unicode string.
Editor_GetLineA	Retrieves the ANSI text on the specified line.
Editor_GetLines	Retrieves the number of the lines for the current document.
Editor_GetLineW	Retrieves the Unicode text on the specified line.
Editor_GetMargin	Retrieves the margin size.
Editor_GetModified	Retrieves the modified state of the text.
Editor_GetOutlineLevel	Retrieves the outline level for the specified logical line.
Editor_GetPageSize	Retrieves a page size.
Editor_GetRef	Retrieves the reference number of a specified plug-in.
Editor_GetRedraw	Retrieves the flag that allows changes in EmEditor to be redrawn or prevents changes in EmEditor to be redrawn.
Editor_GetScrollPos	Retrieves the current positions of the scroll bars.
Editor_GetSelEnd	Retrieves the ending character position of the selection.
Editor_GetSelStart	Retrieves the starting character position of the selection.
Editor_GetSelTextA	Retrieves the selected ANSI text.
Editor_GetSelTextW	Retrieves the selected Unicode text.
Editor_GetSelType	Obtains the type of selection status.
Editor_GetSelTypeEx	Obtains the type of selection status.
Editor_GetStatusA	Retrieves the ANSI text displayed on the status bar.
Editor_GetStatusW	Retrieves the Unicode text displayed on the status bar.
Editor_GetVersion	Returns the version number.
Editor_Help	Displays the specified page of the Help.
Editor_Info	Retrieves or sets the value of one of the information parameters used by EmEditor.
Editor_InsertA	Inserts an ANSI string into the current cursor position.
Editor_InsertFileA	Inserts the specified file contents at the cursor (ANSI).
Editor_InsertFileW	Inserts the specified file contents at the cursor (Unicode).
Editor_InsertStringA	Inserts an ANSI string at the current cursor position. This may overwrite the existing string depending on the current properties.
Editor_InsertStringW	Inserts a Unicode string into the current cursor position. This may overwrite the existing string depending on the current Properties.
Editor_InsertW	Inserts a Unicode string at the current cursor position.
Editor_IsCharHalfOrFull	Determines whether a specified character is a half-width or full-width character.
Editor_KeyboardProp	Displays the Keyboard Properties for the specified command ID and configuration.
Editor_LineFromChar	Retrieves the index of the line that contains the specified character index (the serial position).
Editor_LineIndex	Retrieves the character index of the first character of a specified line in EmEditor.
Editor_LoadConfigA	Reloads a configuration which is specified by name as an ANSI string.
Editor_LoadConfigW	Reloads a configuration which is specified by name as a Unicode string.
Editor_LoadFileA	Loads a specified file into EmEditor (ANSI).

Editor_LoadFileW	Loads a specified file into EmEditor (Unicode).
Editor_LogicalToSerial	Converts the logical coordinates to the serial position.
Editor_LogicalToView	Converts the logical coordinates to the display coordinates.
Editor_MatchRegex	Determines whether a string matches a specified regular expression.
Editor_OutputDir	Sets the current directory for the output bar.
Editor_OutputString	Appends a string to the output bar.
Editor_OverwriteA	Inserts an ANSI string by overwriting the existing string at the current cursor position.
Editor_OverwriteW	Inserts a Unicode string by overwriting the existing string at the current cursor position.
Editor_QueryStatus	Queries the status of the command, whether the command is enabled, and whether the status has been checked.
Editor_Redraw	Allows changes in EmEditor to be redrawn, or prevents changes in EmEditor from being redrawn.
Editor_RegQueryValue	Queries a value from the Registry or an INI file depending on the EmEditor settings.
Editor_RegSetValue	Sets a value into the Registry or an INI file depending on the EmEditor settings.
Editor_Release	Decrements the reference number of the plug-in.
Editor_ReplaceA	Replaces an ANSI string.
Editor_ReplaceW	Replaces an Unicode string.
Editor_ReplaceInFilesA	Replaces an ANSI string in multiple files in the specified location.
Editor_ReplaceInFilesW	Replaces a Unicode string in multiple files in the specified location.
Editor_SaveFileA	Saves the text to a specified file (ANSI).
Editor_SaveFileW	Saves the text to a specified file (Unicode).
Editor_SerialToLogical	Converts the serial position to the logical coordinates.
Editor_SetAnchorPos	Sets the origin point of the selection.
Editor_SetCaretPos	Moves the cursor position.
Editor_SetCaretPosEx	Moves the cursor position and optionally extends the selection.
Editor_SetConfigA	Changes to a configuration specified by an ANSI string.
Editor_SetConfigW	Changes to a configuration specified by a Unicode string.
Editor_SetModified	Changes the modified state of the text.
Editor_SetOutlineLevel	Sets the outline level for the specified logical line.
Editor_SetScrollPos	Specifies the scroll bars position.
Editor_SetScrollPosEx	Specifies the scroll bars position.
Editor_SetSelLength	Changes the character length of the selection.
Editor_SetSelType	Sets the type of selection status.
Editor_SetSelTypeEx	Sets the type of selection status.
Editor_SetSelView	Changes the the starting and ending position of the selection.
Editor_SetStatusA	Displays an ANSI message on the status bar.
Editor_SetStatusW	Displays a Unicode message on the status bar.
Editor_ShowOutline	Shows or hides the outline.
Editor_ToolbarClose	Closes a custom toolbar.
Editor_ToolbarOpen	Opens a custom toolbar.
Editor_ToolbarShow	Shows or hides a custom toolbar.
Editor_UpdateToolbar	Updates a button status in a toolbar.
Editor_ViewToDev	Converts the display coordinates of a specified position to the device (client) coordinates.
Editor_ViewToLogical	Converts the display coordinates of a specified position to the logical coordinates.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_AddRef

Increments the reference number of the plug-in. You can use this inline function or explicitly send the EE_ADD_REF message.

Editor_AddRef(HWND hwnd, HINSTANCE hInstance);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

hInstance

Specifies the instance handle for the plug-in.

Return Values

The return value is the reference number of the plug-in after incremented. If the return value is less than or equal to zero, the specified plug-in can be safely unloaded from EmEditor.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_Convert

Converts characters. You can use this inline function or explicitly send the EE_CONVERT message.

Editor_Convert(HWND hwnd, UINT nFlags);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nFlags

You can specify a combination of the following values.

Value	Meaning
FLAG_MAKE_LOWER	Converts to lowercase characters.
FLAG_MAKE_UPPER	Converts to uppercase characters.
FLAG_HAN_TO_ZEN	Converts to full-size characters.
FLAG_ZEN_TO_HAN	Converts to half-size characters.
FLAG_CAPITALIZE	Capitalizes the first letter of each word.
FLAG_CONVERT_SELECT_ALL	Converts the entire text. If this flag is not set, EE_CONVERT converts the characters only in the selection.
FLAG_CONVERT_KATA	Converts Katakana.
FLAG_CONVERT_ALPHANUMERIC	Converts Alphabets and numeric characters.
FLAG_CONVERT_MARK	Converts marks.
FLAG_CONVERT_SPACE	Converts spaces.
FLAG_CONVERT_KANA_MARK	Converts Kana marks.

Return Values

If the message succeeds, the return value is nonzero. If the message fails, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_CustomBarOpen

Opens a custom bar. If a custom bar is already opened before sending this message, EmEditor closes the custom bar, and opens a new custom bar. You can use this inline function or explicitly send the EE_CUSTOM_BAR_OPEN message.

Editor_CustomBarOpen(HWND hwnd, CUSTOM_BAR_INFO* pCustomBarInfo);

Parameters

pCustomBarInfo

Pointer to the CUSTOM_BAR_INFO structure.

Return Values

The return value is a custom bar ID, which is necessary when the custom bar is closed with the Editor_CustomBarClose inline function. If the message fails, the return value is zero.

Version

Supported on EmEditor Professional Version 6.00 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_CustomBarClose

Closes a custom bar. You can use this inline function or explicitly send the EE_CUSTOM_BAR_CLOSE message.

Editor_CustomBarClose(HWND hwnd, UINT nCustomBarID);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nCustomBarID

Specifies the custom bar to close. This is the return value from the Editor_CustomBarOpen inline function.

Return Values

If the message succeeds, the return value is TRUE. If the message fails, the return value is FALSE.

Version

Supported on EmEditor Professional Version 6.00 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_DocGetConfigA

Retrieves the selected configuration name for the specified document as an ANSI string. You can use this inline function or explicitly send the EE_GET_CONFIGA message.

Editor_DocGetConfigA(HWND hwnd, int iDoc, LPSTR szConfigName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

iDoc

Specifies the index of the target document. If -1 is specified, the currently active document will be targeted.

szConfigName

Specifies a buffer that will receive the configuration name. The buffer size must be at least MAX_CONFIG_NAME bytes.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 5.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_DocGetConfigW

Retrieves the selected configuration name for the specified document as a Unicode string. You can use this inline function or explicitly send the EE_GET_CONFIGW message.

Editor_DocGetConfigW(HWND hwnd, int iDoc, LPWSTR szConfigName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

iDoc

Specifies the index of the target document. If -1 is specified, the currently active document will be targeted.

szConfigName

Specifies a buffer that will receive the configuration name. The buffer size must be at least MAX_CONFIG_NAME in words.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 5.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_DocGetLines

Retrieves the number of the lines for the specified document. You can use this inline function or explicitly send the EE_GET_LINES message.

Editor_GetLines(HWND hwnd, int iDoc, int nLogical);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

iDoc

Specifies the index of the target document. If -1 is specified, the currently active document will be targeted.

nLogical

Specifies one of the following Values.

Value	Meaning

POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

Return Values

Returns the number of the lines in EmEditor. If the last line is ends with a return, the line will be counted. If the text is empty, returns one.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_DocGetModified

Retrieves the modified state of the text of the specified document. You can use this inline function or explicitly send the EE_GET_MODIFIED message.

Editor_DocGetModified(HWND hwnd, int iDoc);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

iDoc

Specifies the index of the target document. If -1 is specified, the currently active document will be targeted.

Return Values

If the text is modified, the return value is TRUE. If the text is not modified, the return value is FALSE.

Version

Supported on EmEditor Professional Version 5.00 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_DocInfo

Retrieves or sets the value of one of the information parameters used by EmEditor. You can use this inline function or explicitly send the EE_INFO message.

Editor_DocInfo(HWND hwnd, int iDoc, int nCmd, LPARAM lParam);

Parameters

nCmd

Specifies a parameter to retrieve or set. It will be one of the followings. This parameter can be one of the values from the following table.

nCmd	Meaning	lParam	Return Value
EI_GET_ENCODE	Retrieves the encoding method to save files.	Not used.	(int)nCP The encoding method.
EI_SET_ENCODE	Sets an encoding method to save files.	(UINT)nCP Specifies an encoding method, whose value begins by CODEPAGE_	Not used.

EI_GET_SIGNATURE	Retrieves whether to sign Unicode/UTF-8 files.	Not used.	(BOOL)bSignature TRUE to sign.
EI_SET_SIGNATURE	Sets whether to sign Unicode/UTF-8 files.	(BOOL)bSignature TRUE to sign.	Not used.
EI_GET_FONT_CHARSET	Retrieves the character set to display.	Not used.	(int)nCharset The character set.
EI_SET_FONT_CHARSET	Sets a character set to display.	(int)nCharset Specifies an character set whose value begins by CHARSET_.	Not used.
EI_GET_FONT_CP	Retrieves the code tab used by the font to display.	Not used.	(UINT)nCP The code tab.
EI_GET_INPUT_CP	Retrieves the code tab used by the input languages.	Not used.	(UINT)nCP The code tab.
EI_GET_SHOW_TAG	Retrieves whether to show the tag highlighted.	Not used.	(BOOL)bShowTag TRUE to highlight the tag.
EI_SET_SHOW_TAG	Sets whether to show the tag highlighted.	(BOOL)bShowTag TRUE to highlight the tag.	Not used.
EI_GET_FILE_NAMEA	Retrieves the file name currently opened, in bytes.	(LPSTR)szFileName Specifies a pointer to a buffer to retrieve the file name. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_FILE_NAMEW	Retrieves the file name currently opened, in Unicode.	(LPSTR)szFileName Specifies a pointer to a buffer to retrieve the file name. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_IS_PROPORTIONAL_FONT	Retrieves whether the display font is proportional.	Not used.	(BOOL)bProportionalFont
EI_GET_NEXT_BOOKMARK	Finds the next book mark position.	(int)yLine Specifies an initial logical line to search from. -1 will search from the beginning of the document.	(int)yLine Returns the searched logical line. -1 will be returned if not found.
EI_GET_HILITE_FIND	Retrieves whether searched strings are highlighted.	Not used.	(BOOL)bShowFindHilite
EI_SET_HILITE_FIND	Sets whether searched strings are highlighted.	(BOOL)bShowFindHilite	Not used.
EI_GET_APP_VERSIONA	Retrieves the version name as an ANSI string.	(LPSTR)szVersionName Specifies a pointer to a buffer to retrieve the version string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
	Retrieves the version	(LPWSTR)szVersionName Specifies a pointer to a buffer to retrieve the version string. The buffer	

EI_GET_APP_VERSIONW	name as a Unicode string.	must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_READ_ONLY	Retrieves whether the document is read-only.	Not used.	(BOOL)bReadOnly
EI_IS_WINDOW_COMBINED	Retrieves whether the windows are combined .	Not used.	(BOOL)bCombined
EI_WINDOW_COMBINE	Sets whether the windows are combined .	(BOOL)bCombined Combines the windows if TRUE, or separate the windows if FALSE.	Not used.
EI_IS_UNDO_COMBINED	Retrieves whether an inserted string can be undone at once .	Not used.	(BOOL)bCombined
EI_GET_DOC_COUNT	Retrieves the number of opened documents in the current frame window (EmEditor Professional 5.00 or later only).	Not used.	(int)nCount Returns the number of documents.
EI_INDEX_TO_DOC	Converts a document index to a document handle (EmEditor Professional 5.00 or later only).	Specifies the zero-based index of the document.	(HEEDOC)hDoc Returns the handle to the document.
EI_DOC_TO_INDEX	Converts a document handle to a document index.	Specifies the handle to the document.	(int)nIndex Returns the zero-based index of the document.
EI_ZORDER_TO_DOC	Converts a document z-order to a document handle.	Specifies the zero-based z-order of the document.	(HEEDOC)hDoc Returns the handle to the document.
EI_DOC_TO_ZORDER	Converts a document handle to a document z-order.	Specifies the handle to the document.	(int)nZOrder Returns the zero-based z-order of the document.
EI_GET_ACTIVE_INDEX	Retrieves the index of the active document.	Not used.	(int)nIndex Returns the zero-based index of the document.
EI_SET_ACTIVE_INDEX	Activates a document.	Not used.	(BOOL)bSuccess Returns TRUE if succeeded, or FALSE if failed.
EI_GET_FULL_TITLEA	Retrieves the full title of the document in ANSI string.	(LPSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_FULL_TITLEW	Retrieves the full title of the document in Unicode string.	(LPWSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_SHORT_TITLEA	Retrieves the short title of the document in ANSI string.	(LPSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character	Not used.

		long including the terminating NULL character.	
EI_GET_SHORT_TITLEW	Retrieves the short title of the document in Unicode string.	(LPWSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_SAVE_AS_TITLEA	Retrieves the full title of the document except the asterisk (*) indicating modification in ANSI string.	(LPSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_SAVE_AS_TITLEW	Retrieves the full title of the document except the asterisk (*) indicating modification in Unicode string.	(LPWSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_MOVE_ORDER	Moves the document tab order	Specifies the zero-based index of the destination tab.	Not used.
EI_CLOSE_DOC	Closes the document.	Not used.	(BOOL)bSuccess Returns TRUE if succeeded, or FALSE if failed.
EI_SAVE_DOC	Saves the document. If the document is untitled, the **Save As** dialog box will appear.	Not used.	(BOOL)bSuccess Returns TRUE if succeeded, or FALSE if failed. Selecting Cancel in the **Save As** dialog box when the document is untitled will also return FALSE.
EI_GET_CURRENT_FOLDER	Retrieves the current working folder.	(LPWSTR)szDir Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_IS_LARGE_DOC	Retrieves the flag to indicate whether the document is very large.	Not used.	(BOOL)bLarge Returns TRUE if the document is very large. Otherwise, it returns FALSE.

iDoc

Specifies the zero-based index of the target document. If -1 is specified, the currently active document will be targeted.

lParam

Depends on the parameter specified.

Return Values

Depends on the parameter specified.

Version

Supported on EmEditor Professional Version 5.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_DocSaveFileA

Saves the text of the specified document to a specified file. The file name is specified as an ANSI string. You can use this inline function or explicitly send the EE_SAVE_FILEA message.

Editor_DocSaveFileA(HWND hwnd, int iDoc, BOOL bReplace, LPSTR szFileName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

iDoc

Specifies the zero-based index of the target document. If -1 is specified, the currently active document will be targeted.

bReplace

Specifies TRUE if the text will be saved under a specified name; the file name EmEditor holds and the title shown on the EmEditor Window will be changed. Specifies FALSE if a copy of the text is saved; the file name EmEditor holds will not be changed.

szFileName

Specifies a full path file name in bytes.

Return Values

If it is succeeded, the return value is nonzero. If it isn't succeeded, the return value is zero.

Version

Supported on EmEditor Professional Version 5.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_DocSaveFileW

Saves the text of the specified document to a specified file. The file name is specified as a Unicode string. You can use this inline function or explicitly send the EE_SAVE_FILEW message.

Editor_SaveFileW(HWND hwnd, int iDoc, BOOL bReplace, LPWSTR szFileName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

iDoc

Specifies the index of the target document. If -1 is specified, the currently active document will be targeted.

bReplace

Specifies TRUE if the text will be saved as by a specified name; the file name EmEditor holds and the title shown on the EmEditor

Window will be changed. Specifies FALSE if the copy of the text is saved; the file name EmEditor holds will not be changed.

szFileName

Specifies a full path file name in bytes.

Return Values

If it is succeeded, the return value is nonzero. If it isn't succeeded, the return value is zero.

Version

Supported on EmEditor Professional Version 5.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_DocSetConfigA

Changes the specified document to a configuration specified by an ANSI string. You can use this inline function or explicitly send the EE_SET_CONFIGA message.

Editor_SetConfigA(HWND hwnd, int iDoc, LPCSTR szConfigName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

iDoc

Specifies the index of the target document. If -1 is specified, the currently active document will be targeted.

szConfigName

Specifies a configuration by an ANSI string.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 5.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_DocSetConfigW

Changes the specified document to a configuration specified by a Unicode string. You can use this inline function or explicitly send the EE_SET_CONFIGW message.

Editor_SetConfigW(HWND hwnd, int iDoc, LPCWSTR szConfigName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

iDoc

Specifies the index of the target document. If -1 is specified, the currently active document will be targeted.

szConfigName

Specifies a configuration by a Unicode string.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 5.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_DoIdle

Refreshes the toolbar, the window title, the tab, and others. You can use this inline function or explicitly send the EE_DO_IDLE message.

Editor_DoIdle(HWND hwnd, BOOL bResetTab);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

bResetTab

Resets the tab.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 5.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_DevToView

Convert the device (client) coordinates of a specified position to the display coordinates. You can use this inline function or explicitly send the EE_DEV_TO_VIEW message.

Editor_DevToView(HWND hwnd, POINT_PTR* pptDev, POINT_PTR* pptView);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pptDev

Pointer to a POINT_PTR structure that specifies the device coordinates to be converted.

pptView

Pointer to a POINT_PTR structure to receive the converted display coordinates.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_EnumConfig

Enumerates available configurations. You can use this inline function or explicitly send the EE_ENUM_CONFIG message.

Editor_EnumConfig(HWND hwnd, LPWSTR pBuf, size_t cchBuf);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

cchBuf

Specifies the size of the buffer in characters. Note that two null characters will be added at the end of the list of configurations. If 0 is specified, this message returns the size necessary to retrieve the list of configurations.

pBuf

Specifies the pointer to the buffer to retrieve the list of configurations. In this buffer, the list of available configurations each separated by a null character will be retrieved. Two null characters will be added at the end of the list of configurations. If 0 is specified, pBuf can be NULL.

Return Values

The return value is the size necessary to retrieve the list of configurations.

Version

Supported on Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_EnumHighlight

Enumerates highlighted strings. You can use this inline function or explicitly send the EE_ENUM_HIGHLIGHT message.

Editor_EnumHighlight(HWND hwnd, LPWSTR pBuf, size_t cchBuf);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

cchBuf

Specifies the size of the buffer in characters. Note that two null characters will be added at the end of the list of highlighted strings. If 0 is specified, this message returns the size necessary to retrieve the list of highlighted strings.

pBuf

Specifies the pointer to the buffer to retrieve the list of highlighted strings. In this buffer, the list of highlighted strings each separated by a null character will be retrieved. Two null characters will be added at the end of the list of highlighted strings. If 0 is specified, pBuf can be NULL.

The first character of each string represents the color and a combination of the following values.

From 0 to 9	color. Use HIGHLIGHT_COLOR_MASK for mask.
HIGHLIGHT_WORD	whole world only.
HIGHLIGHT_RIGHT_SIDE	highlight right side.
HIGHLIGHT_INSIDE_TAG	inside tag only.
HIGHLIGHT_REG_EXP	regular expression.
HIGHLIGHT_CASE	match case.
HIGHLIGHT_RIGHT_ALL	highlight right all.

Return Values

The return value is the size necessary to retrieve the list of configurations.

Version

Supported on Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_EmptyUndoBuffer

Empties the buffer for the Undo and Redo commands. You can use this inline function or explicitly send the EE_EMPTY_UNDO_BUFFER message.

Editor_EmptyUndoBuffer(HWND hwnd);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_ExecCommand

Executes a specified command. You can use this inline function or explicitly send the EE_EXEC_COMMAND message.

Editor_ExecCommand(HWND hwnd, UINT nCmdID);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nCmdID

The identifier of the command to be executed. See Command IDs.

Return Value

The return value is not used.

Send feedback on this topic to Emurasoft

Editor_FindA

Searches an ANSI string. You can use this inline function or explicitly send the EE_FINDA message.

Editor_FindA(HWND hwnd, UINT nFlags, LPCSTR szFind)

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nFlags

You can specify a combination of the following values.

Value	Meaning
FLAG_FIND_NEXT	Searches the string downward from the cursor position. If this flag is not set, searches the string upward.
FLAG_FIND_CASE	Matches cases.
FLAG_FIND_ESCAPE	Uses escape sequences.
FLAG_FIND_NO_PROMPT	Suppresses displaying a dialog box even if no string is found.
FLAG_FIND_ONLY_WORD	Searches only words.
FLAG_FIND_AROUND	Moves to start/end of the text.
FLAG_FIND_REG_EXP	Uses a regular expression.
FLAG_FIND_CLOSE	Closes the dialog box after finished.

szFind

Specifies a string to search.

Return Values

If the message succeeds, the return value is nonzero. If the message fails, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_FindInFilesA

Searches for an ANSI string in multiple files in the specified location. The list of searched files will be displayed in the current window. If the current document is modified, a message prompt will ask to save the changes to the current file. You can use this inline function or explicitly send the EE_FIND_IN_FILESA message.

Editor_FindInFilesA(HWND hwnd, GREP_INFOA pGrepInfo);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pGrepInfo

Specifies a pointer to the GREP_INFOA Structure.

Return Value

Returns FALSE if the user aborts, or TRUE if not.

Version

Supported on EmEditor Professional Version 4.02 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_FindInFilesW

Searches for a Unicode string in multiple files in the specified location. The list of searched files will be displayed in the current window. If the current document is modified, a message prompt will ask to save the changes to the current file. You can use this inline function or explicitly send the EE_FIND_IN_FILESW message.

Editor_FindInFilesW(HWND hwnd, GREP_INFOW pGrepInfo);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pGrepInfo

Specifies a pointer to the GREP_INFOW Structure.

Return Value

Returns FALSE if the user aborts, or TRUE if not.

Version

Supported on EmEditor Professional Version 4.02 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_FindRegex

Searches a string for a regular expression. You can use this inline function or explicitly send the EE_FIND_REGEX message.

Editor_FindRegex(HWND hwnd, FIND_REGEX_INFO* pFindRegexInfo);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pFindRegexInfo

Pointer to the FIND_REGEX_INFO structure.

Return Values

If a string that matches the specified regular expression is found, the return value is TRUE. If the specified regular expression is not found, the return value is FALSE. If the regular expression has an syntax error or another fatal error occurs, the return value is -1.

Version

Supported on Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_FindW

Searches an Unicode string. You can use this inline function or explicitly send the EE_FINDW message.

Editor_FindW(HWND hwnd, UINT nFlags, LPCWSTR szFind)

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nFlags

You can specify a combination of the following values.

Value	Meaning
FLAG_FIND_NEXT	Searches the string downward from the cursor position. If this flag is not set, searches the string upward.
FLAG_FIND_CASE	Matches cases.
FLAG_FIND_ESCAPE	Uses escape sequences.
FLAG_FIND_NO_PROMPT	Suppresses displaying a dialog box even if no string is found.
FLAG_FIND_ONLY_WORD	Searches only words.
FLAG_FIND_AROUND	Moves to start/end of the text.
FLAG_FIND_REG_EXP	Uses a regular expression.
FLAG_FIND_CLOSE	Closes the dialog box after finished.

szFind

Specifies a string to search.

Return Values

If the message succeeds, the return value is nonzero. If the message fails, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_Free

Frees a specified plug-in. You can use this inline function or explicitly send the EE_FREE message.

Editor_Free(HWND hwnd, ATOM atom);

Parameters

hwnd

Specifies the character code by Unicode to be queried.

atom

Specifies the atom of an specified plug-in file name.

Return Values

If the plug-in is freed, the return value is TRUE. If the plug-in is not freed, the return value is FALSE.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetAccelArray

Retrieves the array of the shortcut keys. You can use this inline function or explicitly send the EE_GET_ACCEL_ARRAY message.

Editor_GetAccelArray(HWND hwnd, ACCEL* pAccel, UINT nBufSize);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nBufSize

Specifies the size of the buffer, in ACCEL, that will receive the shortcut key arrays.

pAccel

Specifies the pointer to the buffer that receives the array of the ACCEL structures.

Return Values

The return value is the size of the buffer, in ACCEL, that is needed to receive the shortcut key arrays.

Version

Supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetAnchorPos

Retrieves the origin point of the selection. You can use this inline function or explicitly send the EE_GET_ANCHOR_POS message.

Editor_GetAnchorPos(HWND hwnd, int nLogical, POINT_PTR* pptPos);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nLogical

Specifies one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

pptPos

Pointer to a POINT_PTR structure that will receive the origin point of the selection.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetCaretPos

Retrieves the current cursor position. You can use this inline function or explicitly send the EE_GET_CARET_POS message.

Editor_GetCaretPos(HWND hwnd, int nLogical, POINT_PTR* pptPos);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nLogical

Specifies one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

pptPos

Pointer to a POINT_PTR structure that will receive the current cursor position.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetCmdID

Obtains the Plug-in command ID. You can use this inline function or explicitly send the EE_GET_CMD_ID message.

Editor_GetCmdID(HWND hwnd, HINSTANCE hInstance);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

hInstance

Specifies the Plug-in instance handle.

Return Values

Returns command ID to run this Plug-in.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetConfigA

Retrieves the selected configuration name as an ANSI string. You can use this inline function or explicitly send the EE_GET_CONFIGA message.

Editor_GetConfigA(HWND hwnd, LPSTR szConfigName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szConfigName

Specifies a buffer that will receive the configuration name. The buffer size must be at least MAX_CONFIG_NAME bytes.

Return Values

The return value is not used.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetConfigW

Retrieves the selected configuration name as a Unicode string. You can use this inline function or explicitly send the EE_GET_CONFIGW message.

Editor_GetConfigW(HWND hwnd, LPWSTR szConfigName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szConfigName

Specifies a buffer that will receive the configuration name. The buffer size must be at least MAX_CONFIG_NAME in words.

Return Values

The return value is not used.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetLineA

Retrieves the ANSI text on the specified line. You can use this inline function or explicitly send the EE_GET_LINEA message.

Editor_GetLineA(HWND hwnd, GET_LINE_INFO* pGetLineInfo, LPSTR szString);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pGetLineInfo

Pointer to the GET_LINE_INFO structure.

szString

Pointer to the buffer that will receive the text.

Return Values

If *pGetLineInfo->cch* is zero, the return value is the required size, in bytes, for a buffer that can receive the text. If *pGetLineInfo->cch*. is not zero, the return value is not used. If *nBufferSize* is not zero, the return value is not used.

Editor_GetLines

Retrieves the number of the lines for the current document. You can use this inline function or explicitly send the EE_GET_LINES message.

Editor_GetLines(HWND hwnd, int nLogical);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nLogical

Specifies one of the following Values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

Return Values

Returns the number of the lines in EmEditor. If the last line is ends with a return, the line will be counted. If the text is empty, returns one.

Editor_GetLineW

Retrieves the Unicode text on the specified line. You can use this inline function or explicitly send the EE_GET_LINEW message.

Editor_GetLineW(HWND hwnd, GET_LINE_INFO* pGetLineInfo, LPWSTR szString);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pGetLineInfo

Pointer to the GET_LINE_INFO structure.

szString

Pointer to the buffer that will receive the text.

Return Values

If *pGetLineInfo->cch* is zero, the return value is the required size, in words, for a buffer that can receive the text. If *pGetLineInfo->cch* is not zero, the return value is not used.

Editor_GetMargin

Retrieves the margin size. You can use this inline function or explicitly send the EE_GET_MARGIN message.

Editor_Convert(HWND hwnd);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

Return Values

Returns the currently selected margin size. If the normal line margin size and the quoted line margin size are different, the larger margin will be returned. If lines are wrapped by the window size, the return value will depend on the current window size.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetModified

Retrieves the modified state of the text. You can use this inline function or explicitly send the EE_GET_MODIFIED message.

Editor_GetModified(HWND hwnd);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

Return Values

If the text is modified, the return value is TRUE. If the text is not modified, the return value is FALSE.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetOutlineLevel

Retrieves the outline level for the specified logical line. You can use this inline function or explicitly send the EE_GET_OUTLINE_LEVEL message.

Editor_GetOutlineLevel(HWND hwnd, INT_PTR nLogicalLine);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nLogicalLine

Specifies a logical line.

Return Values

The return value is the outline level for the specified logical line.

Version

Supported on EmEditor Professional Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetPageSize

Retrieves a page size. You can use this inline function or explicitly send the EE_GET_PAGE_SIZE message.

Editor_GetPageSize(HWND hwnd, SIZE_PTR* psizePage);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

psizePage

Pointer to a SIZE_PTR structure that will receive a page size. The page size is a pair of a number of lines and a number of columns that can display a page in the current EmEditor Window size.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetRedraw

Retrieves the flag that allows changes in EmEditor to be redrawn or prevents changes in EmEditor to be redrawn. You can use this inline function or explicitly send the EE_GET_REDRAW message.

Editor_GetRedraw(HWND hwnd);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 5.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetRef

Retrieves the reference number of a specified plug-in. You can send this inline function or explicitly send the EE_GET_REF message.

Editor_GetRef(HWND hwnd, ATOM atom);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

atom

Specifies the atom of an specified plug-in file name.

Return Values

The return value is the reference number of a specified plug-in. If the return value is zero, the specified plug-in can be safely unloaded from EmEditor.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetScrollPos

Retrieves the current positions of the scroll bars. You can use this inline function or explicitly send the EE_GET_SCROLL_POS message.

Editor_GetScrollPos(HWND hwnd, POINT_PTR* pptPos);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pptPos

Pointer to a POINT_PTR structure that will receive the scroll bar positions.

Return Values

The return value is not used.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetSelEnd

Retrieves the ending character position of the selection. You can use this inline function or explicitly send the EE_GET_SEL_END message.

Editor_GetSelEnd(HWND hwnd, int nLogical, POINT_PTR* pptPos);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nLogical

Specifies one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

pptPos

Pointer to a POINT_PTR structure that will receive the ending character position of the selection.

Return Values

The return value is not used.

Editor_GetSelStart

Retrieves the starting character position of the selection. You can use this inline function or explicitly send the EE_GET_SEL_START message.

Editor_GetSelStart(HWND hwnd, int nLogical, POINT_PTR* pptPos);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nLogical

Specifies one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

pptPos

Pointer to a POINT_PTR structure that will receive the starting character position of the selection.

Return Values

The return value is not used.

Editor_GetSelTextA

Retrieves the selected ANSI text. You can use this inline function or explicitly send the EE_GET_SEL_TEXTA message.

Editor_GetSelTextA(HWND hwnd, UINT nBufferSize, LPSTR szBuffer);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nBufferSize

Specifies the maximum number of characters in bytes to copy to the buffer, including the NULL character.

szBuffer

Pointer to the buffer that will receive the text.

Return Values

If *nBufferSize*. is zero, the return value is the required size, in bytes, for a buffer that can receive the text. If *nBufferSize* is not zero, the return value is not used.

Editor_GetSelTextW

Retrieves the selected Unicode text. You can use this inline function or explicitly send the EE_GET_SEL_TEXTW message.

Editor_GetSelTextW(HWND hwnd, UINT nBufferSize, LPWSTR szBuffer);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nBufferSize

Specifies the maximum number of characters in words to copy to the buffer, including the NULL character.

szBuffer

Pointer to the buffer that will receive the text.

Return Values

If *nBufferSize* is zero, the return value is the required size, in words, for a buffer that can receive the text. If *nBufferSize* is not zero, the return value is not used.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetSelType

Obtains the type of selection status. You can use this inline function or explicitly send the EE_GET_SEL_TYPE message.

Editor_GetSelType(HWND hwnd);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

Return Values

Returns a combination of the following values. SEL_TYPE_NONE, SEL_TYPE_CHAR, SEL_TYPE_LINE, and SEL_TYPE_BOX cannot be combined. Only SEL_TYPE_KEYBOARD can be combined with other values.

Value	Meaning
SEL_TYPE_NONE	None is selected.
SEL_TYPE_CHAR	Characters are selected.
SEL_TYPE_LINE	Lines are selected.
SEL_TYPE_BOX	Boxes are selected.
SEL_TYPE_KEYBOARD	Selected by the keyboard.

Version

Supported on EmEditor Professional Version 3.00 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetSelType

Obtains the type of selection status. You can use this inline function or explicitly send the EE_GET_SEL_TYPE message.

Editor_GetSelType(HWND hwnd, BOOL bNeedAlways);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

bNeedAlways

If this parameter is TRUE, EE_GET_SEL_TYPE returns the type of selection status even if none is selected. If this parameter is FALSE, EE_GET_SEL_TYPE returns SEL_TYPE_NONE if none is selected.

Return Values

Returns a combination of the following values. SEL_TYPE_NONE, SEL_TYPE_CHAR, SEL_TYPE_LINE, and SEL_TYPE_BOX cannot be combined. SEL_TYPE_KEYBOARD and SEL_TYPE_SELECTED can be combined with other values. If bNeedAlways is TRUE and if text is selected, a logical sum with SEL_TYPE_SELECTED will be returned. If bNeedAlways is FALSE, SEL_TYPE_SELECTED will not be used.

Value	Meaning
SEL_TYPE_NONE	None is selected.
SEL_TYPE_CHAR	Characters are selected.
SEL_TYPE_LINE	Lines are selected.
SEL_TYPE_BOX	Boxes are selected.
SEL_TYPE_KEYBOARD	Selected by the keyboard.
SEL_TYPE_SELECTED	Selected (when bNeedAlways = TRUE)

Version

Supported on EmEditor Professional Version 3.00 or later. However, bNeedAlways is supported on Version 5.00 or later. On the previous versions, bNeedAlways is assumed to be FALSE.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_GetStatusA

Retrieves the ANSI text displayed on the status bar. You can use this inline function or explicitly send the EE_GET_STATUSA message.

Editor_GetStatusA(HWND hwnd, LPSTR szStatus, UINT nBufSize);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nBufSize

Specifies the size of buffer in characters to retrieve the string, including the terminating null character. You can specify 0 if *szStatus* is NULL. If the buffer size is not enough, *szStatus* will retrieve no string.

szStatus

Specifies the buffer to retrieve the string. If NULL is specified, returns the size of the buffer enough to retrieve the string.

Return Values

Returns the size of the buffer in characters enough to retrieve the string including the terminating null character.

Send feedback on this topic to Emurasoft

Editor_GetStatusW

Retrieves the Unicode text displayed on the status bar. You can use this inline function or explicitly send the EE_GET_STATUSW message.

Editor_GetStatusW(HWND hwnd, LPWSTR szStatus, UINT nBufSize);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nBufSize

Specifies the size of buffer in characters to retrieve the string, including the terminating null character. You can specify 0 if *szStatus* is NULL. If the buffer size is not enough, *szStatus* will retrieve no string.

szStatus

Specifies the buffer to retrieve the string. If NULL is specified, returns the size of the buffer enough to retrieve the string.

Return Values

Returns the size of the buffer in characters enough to retrieve the string including the terminating null character.

Send feedback on this topic to Emurasoft

Editor_GetVersion

Returns the version number. You can use this inline function or explicitly send the EE_GET_VERSION message.

Editor_GetVersion(HWND hwnd);

Editor_GetVersionEx(HWND hwnd, int* pnProductType);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pnProductType

Specifies a pointer to an integer value. This message returns one of the following value.

VERSION_PRO	EmEditor Professional
VERSION_STD	EmEditor Standard

Return Values

Returns the version number multiplied by 1000.

Send feedback on this topic to Emurasoft

Editor_Help

Displays the specified page of the Help. You can use this inline function or explicitly send the EE_HELP message.

Editor_Help(HWND hwnd, LPCTSTR szPageURL);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szPageURL

Specifies the URL of the Help page to display.

Return Values

Return value is not used.

Version

Supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_Info

Retrieves or sets the value of one of the information parameters used by EmEditor. You can use this inline function or explicitly send the EE_INFO message.

Editor_Info(HWND hwnd, int nCmd, LPARAM lParam);

Parameters

nCmd

Specifies a parameter to retrieve or set. It will be one of the followings. This parameter can be one of the values from the following table.

nCmd	Meaning	lParam	Return Value
EI_GET_ENCODE	Retrieves the encoding method to save files.	Not used.	(int)nCP The encoding method.
EI_SET_ENCODE	Sets an encoding method to save files.	(UINT)nCP Specifies an encoding method, whose value begins by CODEPAGE_	Not used.
EI_GET_SIGNATURE	Retrieves whether to sign Unicode/UTF-8 files.	Not used.	(BOOL)bSignature TRUE to sign.
EI_SET_SIGNATURE	Sets whether to sign Unicode/UTF-8 files.	(BOOL)bSignature TRUE to sign.	Not used.
EI_GET_FONT_CHARSET	Retrieves the character set to display.	Not used.	(int)nCharset The character set.
EI_SET_FONT_CHARSET	Sets a character set to display.	(int)nCharset Specifies an character set whose value begins by CHARSET_.	Not used.
EI_GET_FONT_CP	Retrieves the code tab used by the font to display.	Not used.	(UINT)nCP The code tab.
EI_GET_INPUT_CP	Retrieves the code tab used by the input languages.	Not used.	(UINT)nCP The code tab.
EI_GET_SHOW_TAG	Retrieves whether to show the tag highlighted.	Not used.	(BOOL)bShowTag TRUE to highlight the tag.

EI_SET_SHOW_TAG	Sets whether to show the tag highlighted.	(BOOL)bShowTag TRUE to highlight the tag.	Not used.
EI_GET_FILE_NAMEA	Retrieves the file name currently opened, in bytes.	(LPSTR)szFileName Specifies a pointer to a buffer to retrieve the file name. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_FILE_NAMEW	Retrieves the file name currently opened, in Unicode.	(LPSTR)szFileName Specifies a pointer to a buffer to retrieve the file name. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_IS_PROPORTIONAL_FONT	Retrieves whether the display font is proportional.	Not used.	(BOOL)bProportionalFont
EI_GET_NEXT_BOOKMARK	Finds the next book mark position.	(int)yLine Specifies an initial logical line to search from. -1 will search from the beginning of the document.	(int)yLine Returns the searched logical line. -1 will be returned if not found.
EI_GET_HILITE_FIND	Retrieves whether searched strings are highlighted.	Not used.	(BOOL)bShowFindHilite
EI_SET_HILITE_FIND	Sets whether searched strings are highlighted.	(BOOL)bShowFindHilite	Not used.
EI_GET_APP_VERSIONA	Retrieves the version name as an ANSI string.	(LPSTR)szVersionName Specifies a pointer to a buffer to retrieve the version string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_APP_VERSIONW	Retrieves the version name as a Unicode string.	(LPWSTR)szVersionName Specifies a pointer to a buffer to retrieve the version string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_READ_ONLY	Retrieves whether the document is read-only.	Not used.	(BOOL)bReadOnly
EI_IS_WINDOW_COMBINED	Retrieves whether the windows are combined .	Not used.	(BOOL)bCombined
EI_WINDOW_COMBINE	Sets whether the windows are combined .	(BOOL)bCombined Combines the windows if TRUE, or separate the windows if FALSE.	Not used.
EI_IS_UNDO_COMBINED	Retrieves whether an inserted string can be undone at once .	Not used.	(BOOL)bCombined
EI_GET_DOC_COUNT	Retrieves the number of opened documents in the current frame window (EmEditor	Not used.	(int)nCount Returns the number of documents.

	Professional 5.00 or later only).		
EI_INDEX_TO_DOC	Converts a document index to a document handle (EmEditor Professional 5.00 or later only).	Specifies the zero-based index of the document.	(HEEDOC)hDoc Returns the handle to the document.
EI_DOC_TO_INDEX	Converts a document handle to a document index.	Specifies the handle to the document.	(int)nIndex Returns the zero-based index of the document.
EI_ZORDER_TO_DOC	Converts a document z-order to a document handle.	Specifies the zero-based z-order of the document.	(HEEDOC)hDoc Returns the handle to the document.
EI_DOC_TO_ZORDER	Converts a document handle to a document z-order.	Specifies the handle to the document.	(int)nZOrder Returns the zero-based z-order of the document.
EI_GET_ACTIVE_INDEX	Retrieves the index of the active document.	Not used.	(int)nIndex Returns the zero-based index of the document.
EI_SET_ACTIVE_INDEX	Activates a document.	Not used.	(BOOL)bSuccess Returns TRUE if succeeded, or FALSE if failed.
EI_GET_FULL_TITLEA	Retrieves the full title of the document in ANSI string.	(LPSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_FULL_TITLEW	Retrieves the full title of the document in Unicode string.	(LPWSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_SHORT_TITLEA	Retrieves the short title of the document in ANSI string.	(LPSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_SHORT_TITLEW	Retrieves the short title of the document in Unicode string.	(LPWSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_GET_SAVE_AS_TITLEA	Retrieves the full title of the document except the asterisk (*) indicating modification in ANSI string.	(LPSTR)szTitle Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
	Retrieves the full title of the document	(LPWSTR)szTitle Specifies the buffer to retrieve the string. The	

EI_GET_SAVE_AS_TITLEW	except the asterisk (*) indicating modification in Unicode string.	buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_MOVE_ORDER	Moves the document tab order	Specifies the zero-based index of the destination tab.	Not used.
EI_CLOSE_DOC	Closes the document.	Not used.	(BOOL)bSuccess Returns TRUE if succeeded, or FALSE if failed.
EI_SAVE_DOC	Saves the document. If the document is untitled, the **Save As** dialog box will appear.	Not used.	(BOOL)bSuccess Returns TRUE if succeeded, or FALSE if failed. Selecting Cancel in the **Save As** dialog box when the document is untitled will also return FALSE.
EI_GET_CURRENT_FOLDER	Retrieves the current working folder.	(LPWSTR)szDir Specifies the buffer to retrieve the string. The buffer must be MAX_PATH character long including the terminating NULL character.	Not used.
EI_IS_LARGE_DOC	Retrieves the flag to indicate whether the document is very large.	Not used.	(BOOL)bLarge Returns TRUE if the document is very large. Otherwise, it returns FALSE.

lParam

Depends on the parameter specified.

Return Values

Depends on the parameter specified.

Version

Supported on EmEditor Professional Version 3.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_InsertA

Inserts an ANSI string at the current cursor position. This does not overwrite the existing string. You can use this inline function or explicitly send the EE_INSERT_STRINGA message.

Editor_InsertA(HWND hwnd, LPCSTR szString, bool bKeepDestReturnType = false);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szString

Specifies the string to be inserted.

bKeepDestReturnType

Specifies that the destination return type (CR only, LF only or both CR and LF) should be kept. When this parameter is omitted, EmEditor keeps the return type specified in the szString parameter.

Return Values

The return value is not used.

Version

The bKeepDestReturnType flag is supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_InsertFileA

Inserts the specified file contents at the cursor. The file name is specified as an ANSI string. You can use this inline function or explicitly send the EE_INSERT_FILEA message.

Editor_InsertFileA(HWND hwnd, LOAD_FILE_INFO* pLoadFileInfo, LPCSTR szFileName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pLoadFileInfo

Pointer to a LOAD_FILE_INFO structure. If this parameter is NULL, Editor_InsertFileA will open a file by a method predefined by the properties.

szFileName

Specifies a full path file name. If a non-existing file is specified, Editor_InsertFileA will fail.

Return Values

If the command is successful, the return value is nonzero. If the command it not successful, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_InsertFileW

Inserts the specified file contents at the cursor. The file name is specified as a Unicode string. You can use this inline function or explicitly send the EE_INSERT_FILEW message.

Editor_InsertFileW(HWND hwnd, LOAD_FILE_INFO* pLoadFileInfo, LPCWSTR szFileName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pLoadFileInfo

Pointer to a LOAD_FILE_INFO structure. If this parameter is NULL, Editor_InsertFileW will open a file by a method predefined by the properties.

szFileName

Specifies a full path file name. If a non-existing file is specified, Editor_InsertFileW will fail.

Return Values

If the command is successful, the return value is nonzero. If the command it not successful, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_InsertStringA

Inserts an ANSI string into the current cursor position. This may overwrite the existing string depending on the current Properties. You can use this inline function or explicitly send the EE_INSERT_STRINGA message.

Editor_InsertStringA(HWND hwnd, LPCSTR szString, bool bKeepDestReturnType = false);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szString

Specifies the string to be inserted.

bKeepDestReturnType

Specifies that the destination return type (CR only, LF only or both CR and LF) should be kept. When this parameter is omitted, EmEditor keeps the return type specified in the szString parameter.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_InsertStringW

Inserts a Unicode string into the current cursor position. This may overwrite the existing string depending on the current Properties. You can use this inline function or explicitly send the EE_INSERT_STRINGW message.

Editor_InsertStringW(HWND hwnd, LPCWSTR szString, bool bKeepDestReturnType = false);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szString

Specifies the string to be inserted.

bKeepDestReturnType

Specifies that the destination return type (CR only, LF only or both CR and LF) should be kept. When this parameter is omitted, EmEditor keeps the return type specified in the szString parameter.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_InsertW

Inserts a Unicode string at the current cursor position. This does not overwrite the existing string. You can use this inline function or explicitly send the EE_INSERT_STRINGW message.

Editor_InsertW(HWND hwnd, LPCWSTR szString, bool bKeepDestReturnType = false);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szString

Specifies the string to be inserted.

bKeepDestReturnType

Specifies that the destination return type (CR only, LF only or both CR and LF) should be kept. When this parameter is omitted, EmEditor keeps the return type specified in the szString parameter.

Return Values

The return value is not used.

Version

The bKeepDestReturnType flag is supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_IsCharHalfOrFull

Determines whether a specified character is a half-width or full-width character. You can use this inline function or explicitly send the EE_IS_CHAR_HALF_OR_FULL message.

Editor_IsCharHalfOrFull(HWND hwnd, WCHAR ch);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

ch

Specifies the character code by Unicode to be queried.

Return Values

If the character is a full-width character, the return value is two. If the character is a half-width character, the return value is one.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_KeyboardProp

Displays the Keyboard Properties for the specified command ID and configuration. You can use this inline function or explicitly send the

EE_KEYBOARD_PROP message.

Editor_KeyboardProp(HWND hwnd, UINT nCmdID, LPCWSTR pszConfigName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nCmdID

Specifies the command ID for the initial selection on the Keyboard Properties.

pszConfigName

Specifies the configuration for which EmEditor displays the Keyboard Properties.

Return Values

If the user selects OK on the Configuration Properties, the return value is TRUE. If the user selects Cancel, the return value is FALSE.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_LineFromChar

Retrieves the index of the line that contains the specified character index (the serial position). A character index is the zero-based index of the character from the beginning of the entire text. You can use this inline function or explicitly send the EE_LINE_FROM_CHAR message.

Editor_LineFromChar(HWND hwnd, int nLogical, UINT nSerialIndex);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nLogical

Specify one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

nSerialIndex

Specifies the character index of the character contained in the line whose number is to be retrieved. If this parameter is -1, EE_LINE_FROM_CHAR retrieves the line number of the current line (the line containing the cursor).

Return Values

The return value is the zero-based line number of the line containing the character index specified by *nSerialIndex*.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_LineIndex

Retrieves the character index of the first character of a specified line in EmEditor. A character index is the zero-based index of the character from the beginning of the edit control. You can use this inline function or explicitly send the EE_LINE_INDEX message.

Editor_LineIndex(HWND hwnd, BOOL bLogical, int yLine);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

bLogical

Specifies TRUE if the line number is by the logical coordinates. Specifies FALSE if the line number is by the display coordinates.

yLine

Specifies the zero-based line number. A value of -1 specifies the current line number (the line that contains the cursor).

Return Values

The return value is the character index of the line specified in the *yLine* parameter.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_LoadConfigA

Reloads a configuration which is specified by name as an ANSI string. You can use this inline function or explicitly send the EE_LOAD_CONFIGA message.

Editor_LoadConfigA(HWND hwnd, LPCSTR szConfigName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szConfigName

Specifies the name of a configuration to be reloaded.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_LoadConfigW

Reloads a configuration which is specified by name as a Unicode string. You can use this inline function or explicitly send the EE_LOAD_CONFIGW message.

Editor_LoadConfigW(HWND hwnd, LPCWSTR szConfigName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szConfigName

Specifies the name of a configuration to be reloaded.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_LoadFileA

Loads a specified file into EmEditor. The file name is specified as an ANSI string. You can use this inline function or explicitly send the EE_LOAD_FILEA message.

Editor_LoadFileA(HWND hwnd, LOAD_FILE_INFO* pLoadFileInfo, LPCSTR szFileName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pLoadFileInfo

Pointer to a LOAD_FILE_INFO structure. If this parameter is NULL, Editor_LoadFileA will open a file by a method predefined by the properties.

szFileName

Specifies a full path file name in bytes. If a non-existing file is specified, Editor_LoadFileA will fail.

Return Values

If the command is enable, the return value is nonzero. If the command it not enable, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_LoadFileW

Loads a specified file into EmEditor. The file name is specified as a Unicode string. You can use this inline function or explicitly send the EE_LOAD_FILEW message.

Editor_LoadFileW(HWND hwnd, LOAD_FILE_INFO* pLoadFileInfo, LPCWSTR szFileName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pLoadFileInfo

Pointer to a LOAD_FILE_INFO structure. If this parameter is NULL, Editor_LoadFileW will open a file by a method predefined by the properties.

szFileName

Specifies a full path file name in bytes. If a non-existing file is specified, Editor_LoadFileW will fail.

Return Values

If the command is enable, the return value is nonzero. If the command it not enable, the return value is zero.

Send feedback on this topic to Emurasoft

Editor_LogicalToSerial

Convert the logical coordinates to the serial position. The serial position is the zero-based index of the character from the beginning of the entire text. You can use this inline function or explicitly send the EE_LOGICAL_TO_SERIAL message.

Editor_LogicalToSerial(HWND hwnd, POINT_PTR* pptLogical);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pptLogical

Pointer to a POINT_PTR structure that specifies the logical coordinates to be converted.

Return Values

Return the serial position.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_LogicalToView

Convert the logical coordinates to the display coordinates. You can use this inline function or explicitly send the EE_LOGICAL_TO_VIEW message.

Editor_LogicalToView(HWND hwnd, POINT_PTR* pptLogical, POINT_PTR* pptView);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pptLogical

Pointer to a POINT_PTR structure that specifies the logical coordinates to be converted.

pptView

Pointer to a POINT_PTR structure to receive the converted display coordinates.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_MatchRegex

Determines whether a string matches a specified regular expression. You can use this inline function or explicitly send the EE_MATCH_REGEX message.

Editor_MatchRegex(HWND hwnd, MATCH_REGEX_INFO* pMatchRegexInfo);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pMatchRegexInfo

Pointer to the MATCH_REGEX_INFO structure.

Return Values

If a string matches the specified regular expression, the return value is TRUE. If a string does not match the specified regular expression, the return valud is FALSE. If the regular expression has an syntax error or another fatal error occurs, the return value is -1.

Version

Supported on Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_OutputDir

Sets the current directory for the output bar. You can use this inline function or explicitly send the EE_OUTPUT_DIR message.

Editor_OutputDir(HWND hwnd, LPCWSTR szCurrDir);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szCurrDir

Specifies the current directory. This information is necessary if the text contains a relative path that can be jumped only from the current directory.

Return Values

The return value is not used.

Version

Supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_OutputString

Appends a string to the output bar. You can use this inline function or explicitly send the EE_OUTPUT_STRING message.

Editor_OutputString(HWND hwnd, UINT nFlags, LPCWSTR szString);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szString

Specifies the string to be appended.

nFlags

Specifies a combination of the following values.

FLAG_OPEN_OUTPUT	Opens the output bar.
FLAG_CLOSE_OUTPUT	Closes the output bar.
FLAG_FOCUS_OUTPUT	Sets the keyboard focus to the output bar.
FLAG_CLEAR_OUTPUT	Clears the output bar.

Return Values

If the message succeeds, the return value is nonzero. If the message fails, the return value is zero.

Version

Supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_OverwriteA

Inserts an ANSI string at the current cursor position. This overwrites the existing string. You can use this inline function or explicitly send the EE_INSERT_STRINGA message.

Editor_OverwriteA(HWND hwnd, LPCSTR szString, bool bKeepDestReturnType = false);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szString

Specifies the string to be inserted.

bKeepDestReturnType

Specifies that the destination return type (CR only, LF only or both CR and LF) should be kept. When this parameter is omitted, EmEditor keeps the return type specified in the szString parameter.

Return Values

The return value is not used.

Version

The bKeepDestReturnType flag is supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_OverwriteW

Inserts a Unicode string at the current cursor position. This overwrites the existing string. You can use this inline function or explicitly send the EE_INSERT_STRINGW message.

Editor_OverwriteW(HWND hwnd, LPCWSTR szString, bool bKeepDestReturnType = false);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szString

Specifies the string to be inserted.

bKeepDestReturnType

Specifies that the destination return type (CR only, LF only or both CR and LF) should be kept. When this parameter is omitted, EmEditor keeps the return type specified in the szString parameter.

Return Values

The return value is not used.

Version

The bKeepDestReturnType flag is supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_QueryStatus

Queries the status of the command, whether the command is enabled, and whether the status has been checked. You can use this inline function or explicitly send the EE_QUERY_STATUS message.

Editor_QueryStatus(HWND hwnd, UINT nCmdID, BOOL* pbChecked);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nCmdID

The identifier of the command on which the status is queried. See Command IDs.

pbChecked

Pointer to a variable that receives a status check (TRUE indicates the command is checked, FALSE indicates the command is not checked).

Return Values

If the command is enable, the return value is nonzero. If the command it not enable, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_Redraw

Allows changes in EmEditor to be redrawn or prevents changes in EmEditor from being redrawn. You can use this inline function or explicitly send the EE_REDRAW message.

Editor_Redraw(HWND hwnd, BOOL bRedraw);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

bRedraw

Specifies the redraw state. If this parameter is TRUE, the content can be redrawn after a change. If this parameter is FALSE, the

content cannot be redrawn after a change.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_RegQueryValue

Retrieves the data for the specified value from the Registry or an INI file depending on the EmEditor settings. You can use this inline function or explicitly send the EE_REG_QUERY_VALUE message.

Editor_RegQueryValue(HWND hwnd, DWORD dwKey, LPCWSTR pszConfig, LPCWSTR pszValue, DWORD dwType, BYTE* lpData, DWORD* lpcbData, DWORD dwFlags);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

dwKey

Specifies one of the following values to specify a key. EEREG_CONFIG and EEREG_EMEDITORPLUGIN require the pszConfig parameter to specify the key.

EEREG_COMMON	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Common or eeCommon.ini\[Common]
EEREG_REGIST	HKEY_CURRENT_USER\Software\EmSoft\Regist or eeCommon.ini\[Regist]
EEREG_MACROS	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Macros or eeCommon.ini\[Macros]
EEREG_PLUGINS	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\PlugIns or eeCommon.ini\[PlugIns]
EEREG_RECENT_FILE_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent File List or eeCommon.ini\[Recent File List]
EEREG_RECENT_FOLDER_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Folder List or eeCommon.ini\[Recent Folder List]
EEREG_RECENT_FONT_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Font List or eeCommon.ini\[Recent Font List]
EEREG_RECENT_INSERT_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Insert List or eeCommon.ini\[Recent Insert List]
EEREG_AUTOSAVE	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\AutoSave or eeCommon.ini\[AutoSave]
EEREG_LM_COMMON	HKEY_LOCAL_MACHINE\SOFTWARE\EmSoft\EmEditor v3\Common or eeLM.ini\[Common]
EEREG_LM_REGIST	HKEY_LOCAL_MACHINE\SOFTWARE\EmSoft\Regist or eeLM.ini\[Regist]
EEREG_CONFIG	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3 \Config\(pszConfig) or eeConfig.ini\[(pszConfig)]
EEREG_EMEDITORPLUGIN	HKEY_CURRENT_USER\Software\EmSoft\EmEditorPlugIns\(pszConfig) or eePlugin.ini\[(pszConfig)]

pszConfig

Specifies an additional string to specify the key when EEREG_CONFIG or EEREG_EMEDITORPLUGIN is selected.

pszValue

Specifies the name of the value to be retrieved.

dwType

Specifies one of the following values to specify the type of data pointed to by the lpData parameter.

REG_BINARY	Binary data in any form.
REG_DWORD	A 32-bit number.
REG_SZ	A null-terminated Unicode string.

lpData

A pointer to a buffer that receives the value's data. This parameter can be NULL only if the data is of type REG_BINARY.

lpcbData

A pointer to a variable that specifies the size of the buffer pointed to by the lpData parameter, in bytes. When the function returns, this variable contains the size of the data copied to lpData.

dwFlags

This parameter is reserved and must be zero.

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value is a nonzero error code defined in Winerror.h.

Version

Supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_RegSetValue

Sets a value into the Registry or an INI file depending on the EmEditor settings. You can use this inline function or explicitly send the EE_REG_SET_VALUE message.

Editor_RegSetValue(HWND hwnd, DWORD dwKey, LPCWSTR pszConfig, LPCWSTR pszValue, DWORD dwType, const BYTE* lpData, DWORD cbData, DWORD dwFlags);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

dwKey

Specifies one of the following values to specify a key. EEREG_CONFIG and EEREG_EMEDITORPLUGIN require the pszConfig parameter to specify the key.

EEREG_COMMON	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Common or eeCommon.ini\[Common]
EEREG_REGIST	HKEY_CURRENT_USER\Software\EmSoft\Regist or eeCommon.ini\[Regist]
EEREG_MACROS	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Macros or eeCommon.ini\[Macros]
EEREG_PLUGINS	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\PlugIns or eeCommon.ini\[PlugIns]
EEREG_RECENT_FILE_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent File List or eeCommon.ini\[Recent File List]
EEREG_RECENT_FOLDER_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Folder List or eeCommon.ini\[Recent Folder List]
EEREG_RECENT_FONT_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Font List or eeCommon.ini\[Recent Font List]

EEREG_RECENT_INSERT_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Insert List or eeCommon.ini\[Recent Insert List]
EEREG_AUTOSAVE	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\AutoSave or eeCommon.ini\[AutoSave]
EEREG_LM_COMMON	HKEY_LOCAL_MACHINE\SOFTWARE\EmSoft\EmEditor v3\Common or eeLM.ini\[Common]
EEREG_LM_REGIST	HKEY_LOCAL_MACHINE\SOFTWARE\EmSoft\Regist or eeLM.ini\[Regist]
EEREG_CONFIG	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3 \Config\(pszConfig) or eeConfig.ini\[(pszConfig)]
EEREG_EMEDITORPLUGIN	HKEY_CURRENT_USER\Software\EmSoft\EmEditorPlugIns\(pszConfig) or eePlugin.ini\[(pszConfig)]

pszConfig

Specifies an additional string to specify the key when EEREG_CONFIG or EEREG_EMEDITORPLUGIN is selected.

pszValue

Specifies the name of the value to be set. If this parameter is NULL and the dwType parameter is REG_SZ, the entire key pointed to by dwKey and pszConfig parameters, including all entries within the key, is deleted.

dwType

Specifies one of the following values to specify the type of data pointed to by the lpData parameter.

REG_BINARY	Binary data in any form.
REG_DWORD	A 32-bit number.
REG_SZ	A null-terminated Unicode string.

lpData

The data to be stored. For the REG_SZ type, the string must be null-terminated. If this parameter is NULL, the value pointed to by pszValue parameter is removed.

cbData

The size of the information pointed to by the lpData parameter, in bytes. If the data is of type REG_SZ, cbData must include the size of the terminating null character.

dwFlags

This parameter can be EE_REG_VARIABLE_SIZE if the binary data is of a variable size. Otherwise, it must be zero.

Return Values

If the function succeeds, the return value is ERROR_SUCCESS.

If the function fails, the return value is a nonzero error code defined in Winerror.h.

Version

Supported on EmEditor Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_Release

Decrements the reference number of the plug-in. You can use this inline function or explicitly send the EE_RELEASE message.

Editor_Release(HWND hwnd, HINSTANCE hInstance);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

hInstance

Specifies the instance handle for the plug-in.

Return Values

The return value is the reference number of the plug-in after it's been decremented. If the return value is less than or equal to zero, the specified plug-in can be safely unloaded from EmEditor.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_ReplaceA

Replaces an ANSI string. You can use this inline function or explicitly send the EE_REPLACEA message.

Editor_ReplaceA(HWND hwnd, UINT nFlags, LPCSTR szFindReplace);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nFlags

You can specify a combination of the following values.

Value	Meaning
FLAG_FIND_CASE	Matches case.
FLAG_FIND_ESCAPE	Uses escape sequences.
FLAG_REPLACE_SEL_ONLY	Replaces only in the selection when specified with FLAG_REPLACE_ALL.
FLAG_REPLACE_ALL	Replaces all occurrences.
FLAG_FIND_NO_PROMPT	Suppresses displaying a dialog box even if no string is found.
FLAG_FIND_ONLY_WORD	Searches only words.
FLAG_FIND_REG_EXP	Uses a regular expression.
FLAG_FIND_CLOSE	Closes the dialog box after finished.

szFindReplace

Specifies a string to search and a string to replace. You must specify the string to search and the string to replace, in that order. The null character ('\0') must be inserted between the two.

Return Values

If the message succeeds, the return value is nonzero. If the message fails, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_ReplaceInFilesA

Replaces an ANSI string in multiple files in the specified location. You can use this inline function or explicitly send the EE_REPLACE_IN_FILESA message.

Editor_ReplaceInFilesA(HWND hwnd, GREP_INFOA pGrepInfo);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pGrepInfo

Specifies a pointer to the GREP_INFOW Structure.

Return Value

Returns FALSE if the user aborts, or TRUE if not.

Version

Supported on EmEditor Professional Version 4.02 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_ReplaceInFilesW

Replaces a Unicode string in multiple files in the specified location. You can use this inline function or explicitly send the EE_REPLACE_IN_FILESW message.

Editor_ReplaceInFilesW(HWND hwnd, GREP_INFOW pGrepInfo);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pGrepInfo

Specifies a pointer to the GREP_INFOW Structure.

Return Value

Returns FALSE if the user aborts, or TRUE if not.

Version

Supported on EmEditor Professional Version 4.02 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_ReplaceW

Replaces a Unicode string. You can use this inline function or explicitly send the EE_REPLACEW message.

Editor_ReplaceW(HWND hwnd, UINT nFlags, LPCWSTR szFindReplace);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nFlags

You can specify a combination of the following values.

Value	Meaning

FLAG_FIND_CASE	Matches case.
FLAG_FIND_ESCAPE	Uses escape sequences.
FLAG_REPLACE_SEL_ONLY	Replaces only in the selection when specified with FLAG_REPLACE_ALL.
FLAG_REPLACE_ALL	Replaces all occurrences.
FLAG_FIND_NO_PROMPT	Suppresses displaying a dialog box even if no string is found
FLAG_FIND_ONLY_WORD	Searches only words.
FLAG_FIND_REG_EXP	Uses a regular expression.
FLAG_FIND_CLOSE	Closes the dialog box after finished.

szFindReplace

Specifies a string to search and a string to replace. You must specify the string to search and the string to replace, in that order. The null character ('\0') must be inserted between the two.

Return Values

If the message succeeds, the return value is nonzero. If the message fails, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SaveFileA

Saves the text to a specified file. The file name is specified as an ANSI string. You can use this inline function or explicitly send the EE_SAVE_FILEA message.

Editor_SaveFileA(HWND hwnd, BOOL bReplace, LPSTR szFileName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

bReplace

Specifies TRUE if the text will be saved under a specified name; the file name EmEditor holds and the title shown on the EmEditor Window will be changed. Specifies FALSE if a copy of the text is saved; the file name EmEditor holds will not be changed.

szFileName

Specifies a full path file name in bytes.

Return Values

If it is succeeded, the return value is nonzero. If it isn't succeeded, the return value is zero.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SaveFileW

Saves the text to a specified file. The file name is specified as a Unicode string. You can use this inline function or explicitly send the EE_SAVE_FILEW message.

Editor_SaveFileW(HWND hwnd, BOOL bReplace, LPWSTR szFileName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

bReplace

Specifies TRUE if the text will be saved as by a specified name; the file name EmEditor holds and the title shown on the EmEditor Window will be changed. Specifies FALSE if the copy of the text is saved; the file name EmEditor holds will not be changed.

szFileName

Specifies a full path file name in bytes.

Return Values

If it is succeeded, the return value is nonzero. If it isn't succeeded, the return value is zero.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SerialToLogical

Convert the serial position to the logical coordinates. The serial position is the zero-based index of the character from the beginning of the entire text. You can use this inline function or explicitly send the EE_SERIAL_TO_LOGICAL message.

Editor_SerialToLogical(HWND hwnd, UINT nSerial, POINT_PTR* pptLogical);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nSerial

Specifies a serial position to be converted.

pptLogical

Pointer to a POINT_PTR structure that will receive the converted logical coordinates.

Return Values

Return the serial potion.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetAnchorPos

Sets the origin point of the selection. You can use this inline function or explicitly send the EE_SET_ANCHOR_POS message.

Editor_SetAnchorPos(HWND hwnd, int nLogical, POINT_PTR* pptPos);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nLogical

Specifies one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates

POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)

pptPos

Pointer to a POINT_PTR structure that specified the origin point of the selection.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetCaretPos

Specifies the cursor position. You can use this inline function or explicitly send the EE_SET_CARET_POS message.

Editor_SetCaretPos(HWND hwnd, int nLogical, POINT_PTR* pptPos);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nLogical

Specifies one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)
POS_SCROLL_DONT_CARE	The cursor position becomes where the scrolling becomes minimum.
POS_SCROLL_CENTER	The cursor position becomes near the center of the window.
POS_SCROLL_TOP	The cursor position becomes the top of the window.

pptPos

Pointer to a POINT_PTR structure that specified the cursor position.

Return Values

The return value is not used.

Version

Supported on Version 4.03 or later. However, POS_SCROLL_DONT_CARE, POS_SCROLL_CENTER, and POS_SCROLL_TOP flags are supported on Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetCaretPosEx

Moves the cursor position and optionally extends the selection. You can use this inline function or explicitly send the EE_SET_CARET_POS message.

Editor_SetCaretPosEx(HWND hwnd, int nLogical, POINT_PTR* pptPos, BOOL bExtend);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nLogical

Specifies one of the following values.

Value	Meaning
POS_VIEW	Display Coordinates
POS_LOGICAL_A	Logical Coordinates (Count double-byte characters as two)
POS_LOGICAL_W	Logical Coordinates (Count double-byte characters as one)
POS_SCROLL_DONT_CARE	The cursor position becomes where the scrolling becomes minimum.
POS_SCROLL_CENTER	The cursor position becomes near the center of the window.
POS_SCROLL_TOP	The cursor position becomes the top of the window.

pptPos

Pointer to a POINT_PTR structure that specified the cursor position.

bExtend

Determines whether to extend the current selection. If *bExtend* is TRUE, then the active end of the selection moves to the location while the anchor end remains where it is. Otherwise, both ends are moved to the specified location.

Return Values

The return value is not used.

Version

Supported on Version 4.03 or later. However, POS_SCROLL_DONT_CARE, POS_SCROLL_CENTER, and POS_SCROLL_TOP flags are supported on Version 6.00 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetConfigA

Changes to a configuration specified by an ANSI string. You can use this inline function or explicitly send the EE_SET_CONFIGA message.

Editor_SetConfigA(HWND hwnd, LPCSTR szConfigName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szConfigName

Specifies a configuration by an ANSI string.

Return Values

The return value is not used.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetConfigW

Changes to a configuration specified by a Unicode string. You can use this inline function or explicitly send the EE_SET_CONFIGW message.

Editor_SetConfigW(HWND hwnd, LPCWSTR szConfigName);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szConfigName

Specifies a configuration by a Unicode string.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetModified

Changes the modified state of the text. You can use this inline function or explicitly send the EE_SET_MODIFIED message.

Editor_SetModified(HWND hwnd, BOOL bModified);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

bModified

TRUE to change the state as modified, or FALSE to change the state as unmodified.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetOutlineLevel

Sets the outline level for the specified logical line. You can use this inline function or explicitly send the EE_SET_OUTLINE_LEVEL message.

Editor_SetOutlineLevel(HWND hwnd, INT_PTR nLogicalLine, int nLevel);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nLogicalLine

Specifies a logical line.

nLevel

Specifies an outline level.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetScrollPos

Specifies the scroll bar position. You can use this inline function or explicitly send the EE_SET_SCROLL_POS message.

Editor_SetScrollPos(HWND hwnd, POINT_PTR* pptPos);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pptPos

Pointer to a POINT_PTR structure that specifies the scroll bar positions. The cursor position will not be changed.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetScrollPosEx

Specifies the scroll bar position. You can use this inline function or explicitly send the EE_SET_SCROLL_POS message.

Editor_SetScrollPos(HWND hwnd, POINT* pptPos, BOOL bCanMoveCursor);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pptPos

Pointer to a POINT_PTR structure that specifies the scroll bar positions. The cursor position will not be changed.

bCanMoveCursor

If this parameter is TRUE and if the Move Cursor by Scrolling check box is selected, the cursor position will also move. If this parameter is FALSE, the cursor position will not move.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 5.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetSelLength

Changes the character length of the selection. You can use this inline function or explicitly send the EE_SET_SEL_LENGTH message.

Editor_SetSelLength(HWND hwnd, UINT nLen);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nLen

Specifies the character length of the selection. Returns are always two character length (CR+LF).

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

Copyright © 2003-2007 by Emurasoft, Inc.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetSelType

Sets the type of selection status. You can use this inline function or explicitly send the EE_SET_SEL_TYPE message.

Editor_SetSelType(hwnd, nSelType);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nSelType

You can specify a combination of the following values. However, SEL_TYPE_NONE, SEL_TYPE_CHAR, SEL_TYPE_LINE, SEL_TYPE_BOX cannot be combined. Only SEL_TYPE_KEYBOARD can be combined with SEL_TYPE_NONE, SEL_TYPE_CHAR, SEL_TYPE_LINE, or SEL_TYPE_BOX.

SEL_TYPE_NONE	Not selected.
SEL_TYPE_CHAR	Stream selection mode.
SEL_TYPE_LINE	Line selection mode.
SEL_TYPE_BOX	Box selection mode.
SEL_TYPE_KEYBOARD	Specifies the keyboard selection mode. This value can be combined with another value.

Return Values

Not used.

Send feedback on this topic to Emurasoft

Copyright © 2003-2007 by Emurasoft, Inc.

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetSelTypeEx

Sets the type of selection status. You can use this inline function or explicitly send the EE_SET_SEL_TYPE message.

Editor_SetSelTypeEx(hwnd, bNeedAlways, nSelType);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

bNeedAlways

If this parameter is TRUE, the type of selection status can be set even if none is selected. If this parameter is FALSE, SEL_TYPE_NONE will cancel the selection.

nSelType

You can specify a combination of the following values. However, SEL_TYPE_NONE, SEL_TYPE_CHAR, SEL_TYPE_LINE, SEL_TYPE_BOX cannot be combined. Only SEL_TYPE_KEYBOARD can be combined with SEL_TYPE_NONE, SEL_TYPE_CHAR, SEL_TYPE_LINE, or SEL_TYPE_BOX.

SEL_TYPE_NONE	Not selected.
SEL_TYPE_CHAR	Stream selection mode.
SEL_TYPE_LINE	Line selection mode.
SEL_TYPE_BOX	Box selection mode.
SEL_TYPE_KEYBOARD	Specifies the keyboard selection mode. This value can be combined with another value.

Return Values

Not used.

Version

Supported on EmEditor Professional Version 5.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetSelView

Changes the the starting and ending poition of the selection. You can use this inline function or explicitly send the EE_SET_SEL_VIEW message.

Editor_SetSelView(HWND hwnd, POINT_PTR* pptSelStart, POINT_PTR* pptSelEnd);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pptSelStart

Pointer to a POINT_PTR structure that specifies the starting position of the selection. The position is by display coordinates.

pptSelEnd

Pointer to a POINT_PTR structure that specifies the ending position of the selection. The position is by display coordinates.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetStatusA

Displays an ANSI message on the status bar. You can use this inline function or explicitly send the EE_SET_STATUSA message.

Editor_SetStatusA(HWND hwnd, LPCSTR szStatus);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szStatus

Specifies a massage text to be displayed on the status bar.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_SetStatusW

Displays a Unicode message on the status bar. You can use this inline function or explicitly send the EE_SET_STATUSW message.

Editor_SetStatusW(HWND hwnd, LPCWSTR szStatus);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

szStatus

Specifies a massage text to be displayed on the status bar.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_ShowOutline

Shows or hides the outline. You can use this inline function or explicitly send the EE_SHOW_OUTLINE message.

Editor_ShowOutline(HWND hwnd, WPARAM nFlags);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nFlags

Specifies one of the following values.

Value	Meaning
SHOW_OUTLINE_SHOW	Shows outline.
SHOW_OUTLINE_HIDE	Hides outline.

Return Values

The return value is not used.

Version

Supported on EmEditor Professional Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_ToolbarClose

Closes a custom toolbar. You can use this inline function or explicitly send the EE_TOOLBAR_CLOSE message.

Editor_ToolbarClose(HWND hwnd, UINT nCustomRebarID);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nToolbarID

Specifies the toolbar to close. This is the return value from the EE_TOOLBAR_OPEN message.

Return Values

If the message succeeds and the toolbar state has been changed, the return value is TRUE. If the message fails or the toolbar state has not been changed, the return value is FALSE.

Version

Supported on EmEditor Professional Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_ToolbarOpen

Opens a custom toolbar. You can use this inline function or explicitly send the EE_TOOLBAR_OPEN message.

Editor_ToolbarOpen(HWND hwnd, TOOLBAR_INFO* pToolbarInfo);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pToolbarInfo

Pointer to the TOOLBAR_INFO structure.

Return Values

The return value is a custom toolbar ID. If the message fails, the return value is zero.

Version

Supported on EmEditor Professional Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_ToolbarShow

Shows or hides a custom toolbar. You can use this inline function or explicitly send the EE_TOOLBAR_SHOW message.

Editor_ToolbarShow(HWND hwnd, UINT nCustomRebarID, BOOL bVisible);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nToolbarID

Specifies the toolbar to close. This is the return value from the EE_TOOLBAR_OPEN message.

bVisible

Specifies TRUE if the toolbar should be visible, or FALSE if the toolbar should be hidden.

Return Values

If the message succeeds and the toolbar state has been changed, the return value is TRUE. If the message fails or the toolbar state has not been changed, the return value is FALSE.

Version

Supported on EmEditor Professional Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_UpdateToolbar

Updates a button status in a toolbar. You can use this inline function or explicitly send the EE_UPDATE_TOOLBAR message.

Editor_UpdateToolbar(HWND hwnd, UINT nCmdID);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

nCmdID

Specifies the Plug-in instance handle.

Return Values

Return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_ViewToDev

Converts the display coordinates of a specified position to the device (client) coordinates. You can use this inline function or explicitly send the EE_VIEW_TO_DEV message.

Editor_ViewToDev(HWND hwnd, POINT_PTR* pptView, POINT_PTR* pptDev);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pptView

Pointer to a POINT_PTR structure that specifies the display coordinates to be converted.

pptDev

Pointer to a POINT_PTR structure to receive the converted device coordinates.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Inline Functions

Editor_ViewToLogical

Convert the display coordinates of a specified position to the logical coordinates. You can use this inline function or explicitly send the EE_VIEW_TO_LOGICAL message.

Editor_ViewToLogical(HWND hwnd, POINT_PTR* pptView, POINT_PTR* pptLogical);

Parameters

hwnd

Specifies the window handle of the view or frame of EmEditor.

pptView

Pointer to a POINT_PTR structure that specifies the display coordinates to be converted.

pptLogical

Pointer to a POINT_PTR structure that will receive the converted logical coordinates.

Return Values

The return value is not used.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference

Command IDs

Command IDs are used by the EE_EXEC_COMMAND message, the Editor_ExecCommand inline function, the EE_QUERY_STATUS message, and the Editor_QueryStatus inline function.

Command IDs are described in Command Reference.

These constants are defined at the header file (plugin.h).

Events

EVENT_CARET_MOVED	The cursor position has been moved.
EVENT_CHANGE	The text has been altered.
EVENT_CHAR	A character has been inserted. The LOWORD (lParam) represents the inserted Unicode character code.
EVENT_CLOSE	Called immediately before closing EmEditor or the plug-in may be freed. A plug-in should release the resource and make the DLL file available to be removed. The first parameter hwnd of the OnEvents function will be NULL. This event does not mean that the plug-in will actually be freed.
EVENT_CLOSE_FRAME	Called when an EmEditor frame window is being closed. (Supported on EmEditor Version 5.00 or later)
EVENT_CONFIG_CHANGED	The property of the current configuration has been changed.
EVENT_CREATE	Called immediately after starting EmEditor or the plug-in has been loaded. LOWORD (lParam) represents the command ID of the plug-in itself.
EVENT_CREATE_FRAME	Called when a new EmEditor frame window is created. This event is also called when the tab is enabled or disabled. LOWORD(lParam) represents the command ID of the plug-in itself. (Supported on EmEditor Version 5.00 or later)
EVENT_CUSTOM_BAR_CLOSED	Called when a custom bar has been closed. EmEditor calls DestroyWindow() against the client window when the custom bar is closed. lParam represents a pointer to the CUSTOM_BAR_CLOSED_INFO structure. (Supported on EmEditor Version 6.00 or later)
EVENT_CUSTOM_BAR_CLOSING	Called when a custom bar is being closed. lParam represents a pointer to the CUSTOM_BAR_CLOSED_INFO structure. (Supported on EmEditor Version 6.00 or later)
EVENT_DOC_CLOSE	Called when a document is about to close. lParam represents a handle (HEEDOC) to the closing document. (Supported on EmEditor Version 5.00 or later)
EVENT_DOC_SEL_CHANGED	Called when an active document is changed. (Supported on EmEditor Version 5.00 or later)
EVENT_FILE_OPENED	A file has been opened.
EVENT_IDLE	Called when idle. (Supported on EmEditor Version 6.00 or later)
EVENT_KILL_FOCUS	The focus has been lost.
EVENT_MODIFIED	The modified status has be changed.
EVENT_SCROLL	A scroll bar position has been changed.
EVENT_SEL_CHANGED	The selection of the text has been changed.
EVENT_SET_FOCUS	The focus has been set.
EVENT_TAB_MOVED	Called when a tab has been moved.
EVENT_TOOLBAR_CLOSED	Called when a custom toolbar has been closed. Unlike the EVENT_CUSTOM_BAR_CLOSED message, EmEditor does not destroy the client window. lParam represents a pointer to the TOOLBAR_INFO structure. (Supported on EmEditor Version 7.00 or later)
EVENT_TOOLBAR_SHOW	Called when a custom toolbar has been hidden or displayed (when the RBBS_HIDDEN style is toggled). lParam represents a pointer to the TOOLBAR_INFO structure. (Supported on EmEditor Version 7.00 or later)

These events are used as the nEvents parameter by the OnEvents function.

These constants are defined at the header file (plugin.h).

Structures

CUSTOM_BAR_INFO	Used by Editor_CustomBarOpen inline function (EE_CUSTOM_BAR_OPEN message).
CUSTOM_BAR_CLOSE_INFO	Used by EVENT_CUSTOM_BAR_CLOSED event.
FIND_REGEX_INFO	Used by Editor_FindRegex inline function (EE_FIND_REGEX message).
GET_LINE_INFO	Used by Editor_GetLineA and Editor_GetLineW macro (EE_GET_LINEA, EE_GET_LINEW message)
GREP_INFOA	Used by Editor_FindInFilesA inline function, Editor_ReplaceInFilesA inline function (EE_FIND_IN_FILESA message, EE_REPLACE_IN_FILESA message).
GREP_INFOW	Used by Editor_FindInFilesW inline function, Editor_ReplaceInFilesW inline function (EE_FIND_IN_FILESW message, EE_REPLACE_IN_FILESW message).
LOAD_FILE_INFO_EX	Used by Editor_LoadFileA and Editor_LoadFileW inline function (EE_LOAD_FILEA, EE_LOAD_FILEW message)
MATCH_REGEX_INFO	Used by Editor_MatchRegex inline function (EE_MATCH_REGEX message).
POINT_PTR	Used to specify a point position. In 32-bit plug-ins, POINT_PTR is the same as the POINT structure. In 64-bit plug-ins, each field is extended to the 64-bit integer from the 32-bit integer.
REG_QUERY_VALUE_INFO	Used by EE_REG_QUERY_VALUE message.
REG_SET_VALUE_INFO	Used by EE_REG_SET_VALUE message.
SIZE_PTR	Used to specify a size. In 32-bit plug-ins, SIZE_PTR is the same as the SIZE structure. In 64-bit plug-ins, each field is extended to the 64-bit integer from the 32-bit integer.
TOOLBAR_INFO	Used by Editor_ToolbarOpen inline function (EE_TOOLBAR_OPEN message) and events related to custom toolbars.

Send feedback on this topic to Emurasoft

CUSTOM_BAR_INFO

Used by Editor_CustomBarOpen inline function (EE_CUSTOM_BAR_OPEN message).

```
typedef struct _CUSTOM_BAR_INFO {
   size_t cbSize;
   HWND hwndCustomBar;
   HWND hwndClient;
   LPCTSTR pszTitle;
   int iPos;
} CUSTOM_BAR_INFO;
```

Members

cbSize

[in] Size of this data structure, in bytes. Set this member to sizeof(CUSTOM_BAR_INFO) before sending the EE_CUSTOM_BAR_OPEN message.

hwndCustomBar

[out] The handle to the custom bar window will be stored when the EE_CUSTOM_BAR_OPEN message succeeds.

hwndClient

[in] The handle to the Client window.

pszTitle

[in] The string used for the custom bar title.

iPos

[in] The custom bar initial position.

0	The left side of the window.
1	The top of the window.
2	The right side of the window.

3	The bottom of the window.

Version

Supported on EmEditor Professional Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Structures

CUSTOM_BAR_CLOSE_INFO

Used by EVENT_CUSTOM_BAR_CLOSED event.

```
typedef struct _CUSTOM_BAR_CLOSE_INFO {
   UINT nID;
   int iPos;
   DWORD dwFlags;
} CUSTOM_BAR_CLOSE_INFO;
```

Members

nID

[in] The custom bar ID.

iPos

[in] The position of the custom bar immediately before it is closed.

0	The left side of the window.
1	The top of the window.
2	The right side of the window.
3	The bottom of the window.

dwFlags

[out] The reason that the custom bar is closed.

0	The custom bar is closed by a user.
CLOSED_FRAME_WINDOW	The frame window is being closed.
CLOSED_ANOTHER_CUSTOM_BAR	The custom bar is closed because another custom bar is opened.

Version

Supported on EmEditor Professional Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Structures

FIND_REGEX_INFO

Used by Editor_FindRegex inline function (EE_FIND_REGEX message).

```
typedef struct _FIND_REGEX_INFO {
   size_t cbSize; // sizeof( FIND_REGEX_INFO )
   UINT nFlags;
   LPCWSTR pszRegex;
   LPCWSTR pszText;
   LPCWSTR* ppszStart;
   LPCWSTR* ppszEnd;
   LPCWSTR* ppszNext;
} FIND_REGEX_INFO;
```

Members

cbSize

[in] Size of this data structure, in bytes. Set this member to sizeof(FIND_REGEX_INFO) before sending the EE_FIND_REGEX message.

nFlags

[in] Specifies a combination of the following values.

FLAG_FIND_CASE	Matches cases.
FLAG_FIND_ONLY_WORD	Searches only words.

pszRegex

[in] Specifies a regular expression to search for.

pszText

[in] Specifies a string to search.

ppszStart

[out] The pointer at the beginning of the string where the regular expression matches.

ppszEnd

[out] The pointer at the end of the string where the regular expression matches.

ppszEnd

[out] The pointer at the position of the string where the next regular expression search should occur if necessary.

Version

Supported on Version 6.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Structures

GET_LINE_INFO

Used by Editor_GetLineA and Editor_GetLineW inline functions (EE_GET_LINEA and EE_GET_LINEW messages)

```
typedef struct _GET_LINE_INFO {
   UINT cch;
   UINT flags;
   UINT yLine;
} GET_LINE_INFO;
```

Fields

cch

Specifies the maximum number of characters to copy to the buffer (szString parameter of Editor_GetLine macro or lParam of EE_GET_LINE message including the NULL character). If zero is specified, the return value by Editor_GetLine macro or EE_GET_LINE message is the required size, in characters, for a buffer that can receive the text.

flags

The low word of this parameter is a combination of the following values.

FLAG_LOGICAL	Specifies *yLine* field by logical coordinates *yLine*
FLAG_WITH_CRLF	Adds return codes to the text

The high word of this parameter is the index of the target document. A one-based index should be specified at the higher word of flags. If 0 is specified at the higher word of flags, the currently active document will be targeted.

yLine

Specifies a line number of the text to retrieve.

Send feedback on this topic to Emurasoft

GREP_INFOA

Used by Editor_FindInFilesA macro, Editor_ReplaceInFilesA inline functions (EE_FIND_IN_FILESA message, EE_REPLACE_IN_FILESA messages).

```
typedef struct _GREP_INFOA {
   UINT cbSize;
   UINT nCP;
   UINT nFlags;
   LPCSTR pszFind;
   LPCSTR pszReplace;
   LPCSTR pszPath;
   LPCSTR pszBackupPath;
   LPCSTR pszFilesToIgnore;
} GREP_INFOA;
```

Fields

cbSize

Specifies size of (GREP_INFOA).

nCP

Specifies a code tab by which a file is opened.

CODEPAGE_ANSI	Normal ANSI
CODEPAGE_UNICODE	Unicode little endian
CODEPAGE_UNICODE_BIGENDIAN	Unicode big endian
CODEPAGE_UTF8	UTF-8
CODEPAGE_UTF7	UTF-7
CODEPAGE_932	Japanese Shift JIS
CODEPAGE_JIS	Japanese JIS
CODEPAGE_EUC	Japanese EUC
CODEPAGE_AUTO_SJIS_JIS	Converts from Japanese Shift JIS and JIS
CODEPAGE_AUTO_SJIS_JIS_EUC	Converts form Japanese Shift JIS、JIS、EUC
Others	All code pages you can used by system
CODEPAGE_DETECT_UNICODE	Detects Unicode. Can be combined with another value.
CODEPAGE_DETECT_UTF8	Detects UTF-8. Can be combined with another value.
CODEPAGE_DETECT_CHARSET	Detects HTML/XML Charset. Can be combined with another value.
CODEPAGE_DETECT_ALL	Detects all code pages. Can be combined with another value.

nFlags

Specifies a combination of the following values.

FLAG_FIND_CASE	Matches cases.
FLAG_FIND_ESCAPE	Uses escape sequences. Cannot be combined with FLAG_FIND_REG_EXP.
FLAG_FIND_ONLY_WORD	Searches only words.
FLAG_FIND_REG_EXP	Uses a regular expression. Cannot be combined with FLAG_FIND_ESCAPE.
FLAG_FIND_RECURSIVE	Searches in subfolders of the specified path.
FLAG_FIND_FILENAMES_ONLY	Displays files names only.
FLAG_REPLACE_KEEP_OPEN	Keeps the modified files open. Cannot combine with eeReplaceBackup.

	Cannot be combined with FLAG_REPLACE_BACKUP.
FLAG_REPLACE_BACKUP	Saves the backups. Cannot be combined with FLAG_REPLACE_KEEP_OPEN.
FLAG_FIND_IGNORE_FILES	Ignores the files or folders specified by *pszFilesToIgnore*.

pszFind

Specifies a string to search for.

pszReplace

When replacing in files, specifies a string to replace with.

pszPath

Specifies a path to search from. It can include wild cards such as * and ?.

pszBackupPath

When replacing in files, specifies the backup folder if *nFlags* includes FLAG_REPLACE_BACKUP.

pszFilesToIgnore

If *nFlags* includes FLAG_FIND_IGNORE_FILES, specifies the file or folder names to ignore. It can include wild cards such as * and ?. To specify multiple files, use semicolons (;) to separate them.

Version

Supported on EmEditor Professional Version 4.02 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Structures

GREP_INFOW

Used by Editor_FindInFilesW macro, Editor_ReplaceInFilesW macro (EE_FIND_IN_FILESW message, EE_REPLACE_IN_FILESW message).

```
typedef struct _GREP_INFOW {
   UINT cbSize;
   UINT nCP;
   UINT nFlags;
   LPCWSTR pszFind;
   LPCWSTR pszReplace;
   LPCWSTR pszPath;
   LPCWSTR pszBackupPath;
   LPCWSTR pszFilesToIgnore;
} GREP_INFOW;
```

Fields

cbSize

Specifies size of (GREP_INFOA).

nCP

Specifies a code tab by which a file is opened.

CODEPAGE_ANSI	Normal ANSI
CODEPAGE_UNICODE	Unicode little endian
CODEPAGE_UNICODE_BIGENDIAN	Unicode big endian
CODEPAGE_UTF8	UTF-8
CODEPAGE_UTF7	UTF-7
CODEPAGE_932	Japanese Shift JIS
CODEPAGE_JIS	Japanese JIS

CODEPAGE_EUC	Japanese EUC
CODEPAGE_AUTO_SJIS_JIS	Converts from Japanese Shift JIS and JIS
CODEPAGE_AUTO_SJIS_JIS_EUC	Converts form Japanese Shift JIS、JIS、EUC
Others	All code pages you can used by system
CODEPAGE_DETECT_UNICODE	Detects Unicode. Can be combined with another value.
CODEPAGE_DETECT_UTF8	Detects UTF-8. Can be combined with another value.
CODEPAGE_DETECT_CHARSET	Detects HTML/XML Charset. Can be combined with another value.
CODEPAGE_DETECT_ALL	Detects all code pages. Can be combined with another value.

nFlags

Specifies a combination of the following values.

FLAG_FIND_CASE	Matches cases.
FLAG_FIND_ESCAPE	Uses escape sequences. Cannot be combined with FLAG_FIND_REG_EXP.
FLAG_FIND_ONLY_WORD	Searches only words.
FLAG_FIND_REG_EXP	Uses a regular expression. Cannot be combined with FLAG_FIND_ESCAPE.
FLAG_FIND_RECURSIVE	Searches in subfolders of the specified path.
FLAG_FIND_FILENAMES_ONLY	Displays files names only.
FLAG_REPLACE_KEEP_OPEN	Keeps the modified files open. Cannot combine with eeReplaceBackup. Cannot be combined with FLAG_REPLACE_BACKUP.
FLAG_REPLACE_BACKUP	Saves the backups. Cannot be combined with FLAG_REPLACE_KEEP_OPEN.
FLAG_FIND_IGNORE_FILES	Ignores the files or folders specified by *pszFilesToIgnore*.

pszFind

Specifies a string to search for.

pszReplace

When replacing in files, specifies a string to replace with.

pszPath

Specifies a path to search from. It can include wild cards such as * and ?.

pszBackupPath

When replacing in files, specifies the backup folder if *nFlags* includes FLAG_REPLACE_BACKUP.

pszFilesToIgnore

If *nFlags* includes FLAG_FIND_IGNORE_FILES, specifies the file or folder names to ignore. It can include wild cards such as * and ?. To specify multiple files, use semicolons (;) to separate them.

Version

Supported on EmEditor Professional Version 4.02 or later.

EmEditor Home - EmEditor Help - Plug-in Reference - Structures

LOAD_FILE_INFO_EX

Used by Editor_LoadFileA and Editor_LoadFileW inline functions (EE_LOAD_FILEA and EE_LOAD_FILEW messages)

```
typedef struct _LOAD_FILE_INFO_EX {
   UINT cbSize;
   UINT nCP;
   BOOL bDetectUnicode;
```

```
   BOOL bDetectAll;
   BOOL bDetectCharset;
   BOOL bDetectUTF8;
   UINT nFlags;
} LOAD_FILE_INFO_EX;
```

Fields

cbSize

Must be sizeof(LOAD_FILE_INFO_EX).

nCP

Specifies a code tab by which a file is opened.

CODEPAGE_ANSI	Normal ANSI
CODEPAGE_UNICODE	Unicode little endian
CODEPAGE_UNICODE_BIGENDIAN	Unicode big endian
CODEPAGE_UTF8	UTF-8
CODEPAGE_UTF7	UTF-7
CODEPAGE_932	Japanese Shift JIS
CODEPAGE_JIS	Japanese JIS
CODEPAGE_EUC	Japanese EUC
CODEPAGE_AUTO_SJIS_JIS	Converts from Japanese Shift JIS and JIS
CODEPAGE_AUTO_SJIS_JIS_EUC	Converts from Japanese Shift JIS、JIS、EUC
Others	All code pages you can use by system

bDetectUnicode

If TRUE, Unicode will be detected.

bDetectAll

If TRUE, all code pages will be detected.

bDetectCharset

If TRUE, HTML/XML Charset will be detected.

bDetectUTF8

If TRUE, UTF-8 will be detected.

nFlags

Specifies a combination of the following values.

LFI_ALLOW_NEW_WINDOW	Opens a file in a new window.

MATCH_REGEX_INFO

Used by Editor_MatchRegex inline function (EE_MATCH_REGEX message).

```
typedef struct _MATCH_REGEX_INFO {
   size_t cbSize; // sizeof( MATCH_REGEX_INFO )
   UINT nFlags;
   LPCWSTR pszRegex;
   LPCWSTR pszText;
} MATCH_REGEX_INFO;
```

Members

cbSize

[in] Size of this data structure, in bytes. Set this member to sizeof(FIND_REGEX_INFO) before sending the EE_FIND_REGEX message.

nFlags

[in] Specifies a combination of the following values.

FLAG_FIND_CASE	Matches cases.

pszRegex

[in] Specifies a regular expression to search for.

pszText

[in] Specifies a string to search.

Version

Supported on Version 6.00 or later.

POINT_PTR

Used to specify a point position. In 32-bit plug-ins, POINT_PTR is the same as the POINT structure. In 64-bit plug-ins, each field is extended to the 64-bit integer from the 32-bit integer.

```
typedef struct tagPOINT_PTR
{
   LONG_PTR x;
   LONG_PTR y;
} POINT_PTR, *PPOINT_PTR;
```

Fields

x

Specifies an x-axis value.

y

Specifies a y-axis value.

REG_QUERY_VALUE_INFO

Used by EE_REG_QUERY_VALUE message.

```
typedef struct _REG_QUERY_VALUE_INFO {
   size_t cbSize;
   DWORD dwKey;
   LPCWSTR pszConfig;
   LPCWSTR pszValue;
   DWORD dwType;
   BYTE* lpData;
   DWORD* lpcbData;
   DWORD dwFlags;
} REG_QUERY_VALUE_INFO;
```

Members

cbSize

Size of this data structure, in bytes. Set this member to sizeof(REG_QUERY_VALUE_INFO).

dwKey

Specifies one of the following values to specify a key. EEREG_CONFIG and EEREG_EMEDITORPLUGIN require the pszConfig parameter to specify the key.

EEREG_COMMON	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Common or eeCommon.ini\[Common]
EEREG_REGIST	HKEY_CURRENT_USER\Software\EmSoft\Regist or eeCommon.ini\[Regist]
EEREG_MACROS	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Macros or eeCommon.ini\[Macros]
EEREG_PLUGINS	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\PlugIns or eeCommon.ini\[PlugIns]
EEREG_RECENT_FILE_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent File List or eeCommon.ini\[Recent File List]
EEREG_RECENT_FOLDER_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Folder List or eeCommon.ini\[Recent Folder List]
EEREG_RECENT_FONT_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Font List or eeCommon.ini\[Recent Font List]
EEREG_RECENT_INSERT_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Insert List or eeCommon.ini\[Recent Insert List]
EEREG_AUTOSAVE	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\AutoSave or eeCommon.ini\[AutoSave]
EEREG_LM_COMMON	HKEY_LOCAL_MACHINE\SOFTWARE\EmSoft\EmEditor v3\Common or eeLM.ini\[Common]
EEREG_LM_REGIST	HKEY_LOCAL_MACHINE\SOFTWARE\EmSoft\Regist or eeLM.ini\[Regist]
EEREG_CONFIG	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3 \Config\(pszConfig) or eeConfig.ini\[(pszConfig)]
EEREG_EMEDITORPLUGIN	HKEY_CURRENT_USER\Software\EmSoft\EmEditorPlugIns\(pszConfig) or eePlugin.ini\[(pszConfig)]

pszConfig

Specifies an additional string to specify the key when EEREG_CONFIG or EEREG_EMEDITORPLUGIN is selected.

pszValue

Specifies the name of the value to be retrieved.

dwType

Specifies one of the following values to specify the type of data pointed to by the lpData parameter.

REG_BINARY	Binary data in any form.
REG_DWORD	A 32-bit number.
REG_SZ	A null-terminated Unicode string.

lpData

A pointer to a buffer that receives the value's data. This parameter can be NULL only if the data is of type REG_BINARY.

lpcbData

A pointer to a variable that specifies the size of the buffer pointed to by the lpData parameter, in bytes. When the function returns, this variable contains the size of the data copied to lpData.

dwFlags

This parameter is reserved and must be zero.

Version

Supported on Version 7.00 or later.

Send feedback on this topic to Emurasoft

REG_SET_VALUE_INFO

Used by EE_REG_SET_VALUE message.

```
typedef struct _REG_SET_VALUE_INFO {
   size_t cbSize;
   DWORD dwKey;
   LPCWSTR pszConfig;
   LPCWSTR pszValue;
   DWORD dwType;
   const BYTE* lpData;
   DWORD cbData;
   DWORD dwFlags;
} REG_SET_VALUE_INFO;
```

Members

cbSize

Size of this data structure, in bytes. Set this member to sizeof(REG_SET_VALUE_INFO).

dwKey

Specifies one of the following values to specify a key. EEREG_CONFIG and EEREG_EMEDITORPLUGIN require the pszConfig parameter to specify the key.

EEREG_COMMON	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Common or eeCommon.ini\[Common]
EEREG_REGIST	HKEY_CURRENT_USER\Software\EmSoft\Regist or eeCommon.ini\[Regist]
EEREG_MACROS	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Macros or eeCommon.ini\[Macros]
EEREG_PLUGINS	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\PlugIns or eeCommon.ini\[PlugIns]
EEREG_RECENT_FILE_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent File List or eeCommon.ini\[Recent File List]
EEREG_RECENT_FOLDER_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Folder List or eeCommon.ini\[Recent Folder List]
EEREG_RECENT_FONT_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Font List or eeCommon.ini\[Recent Font List]
EEREG_RECENT_INSERT_LIST	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Insert List or eeCommon.ini\[Recent Insert List]
EEREG_AUTOSAVE	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\AutoSave or eeCommon.ini\[AutoSave]
EEREG_LM_COMMON	HKEY_LOCAL_MACHINE\SOFTWARE\EmSoft\EmEditor v3\Common or eeLM.ini\[Common]
EEREG_LM_REGIST	HKEY_LOCAL_MACHINE\SOFTWARE\EmSoft\Regist or eeLM.ini\[Regist]
EEREG_CONFIG	HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3 \Config\(pszConfig) or eeConfig.ini\[(pszConfig)]
EEREG_EMEDITORPLUGIN	HKEY_CURRENT_USER\Software\EmSoft\EmEditorPlugIns\(pszConfig) or eePlugin.ini\[(pszConfig)]

pszConfig

Specifies an additional string to specify the key when EEREG_CONFIG or EEREG_EMEDITORPLUGIN is selected.

pszValue

Specifies the name of the value to be set. If this parameter is NULL and the dwType parameter is REG_SZ, the entire key pointed to by dwKey and pszConfig parameters, including all entries within the key, is deleted.

dwType

Specifies one of the following values to specify the type of data pointed to by the lpData parameter.

REG_BINARY	Binary data in any form.
REG_DWORD	A 32-bit number.
REG_SZ	A null-terminated Unicode string.

lpData

The data to be stored. For the REG_SZ type, the string must be null-terminated.

cbData

The size of the information pointed to by the lpData parameter, in bytes. If the data is of type REG_SZ, cbData must include the size of the terminating null character.

dwFlags

This parameter can be EE_REG_VARIABLE_SIZE if the binary data is of a variable size. Otherwise, it must be zero.

Version

Supported on Version 7.00 or later.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Structures

SIZE_PTR

Used to specify a size. In 32-bit plug-ins, SIZE_PTR is the same as the SIZE structure. In 64-bit plug-ins, each field is extended to the 64-bit integer from the 32-bit integer.

```
typedef struct tagSIZE_PTR
{
   LONG_PTR cx;
   LONG_PTR cy;
} SIZE_PTR, *PSIZE_PTR;
```

Fields

x

Specifies an x-axis value.

y

Specifies a y-axis value.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Structures

TOOLBAR_INFO

Used by Editor_ToolbarOpen inline function (EE_TOOLBAR_OPEN message) and events related to custom toolbars.

```
typedef struct _TOOLBAR_INFO {
   size_t cbSize;
   HWND hwndRebar;
   HWND hwndClient;
```

```
    LPCTSTR pszTitle;
    UINT nMask;
    UINT nID;
    UINT nFlags;
    UINT fStyle;
    UINT cxMinChild;
    UINT cyMinChild;
    UINT cx;
    UINT cxIdeal;
    UINT nBand;
    WORD wPlugInCmdID;
} TOOLBAR_INFO;
```

Members

cbSize

Size of this data structure, in bytes. Set this member to sizeof(TOOLBAR_INFO) before sending the TOOLBAR_INFO message.

hwndRebar

EmEditor stores the handle to the rebar window when the toolbar is created inside the EE_TOOLBAR_OPEN message handler.

hwndClient

Specifies the handle to the client toolbar window.

pszTitle

Specifies a title string for the toolbar.

nMask

Specifies a combination of the following values.

TIM_REBAR	hwndRebar parameter is valid.
TIM_CLIENT	hwndClient parameter is valid.
TIM_TITLE	pszTitle parameter is valid.
TIM_ID	nID parameter is valid.
TIM_FLAGS	nFlags parameter is valid.
TIM_STYLE	fStyle parameter is valid.
TIM_MINCHILD	cxMinChild and cyMinChild parameters are valid.
TIM_CX	cx parameter is valid.
TIM_CXIDEAL	cxIdeal parameter is valid.
TIM_BAND	nBand parameter is valid.
TIM_PLUG_IN_CMD_ID	wPlugInCmdID parameter is valid.

nID

Specifies an ID for the toolbar.

nFlags

The reason that the toolbar is closed.

0	The toolbar is closed by a user.
CLOSED_FRAME_WINDOW	The frame window is being closed.

fStyle

Flags that specify the band style. Include RBBS_HIDDEN to hide the toolbar. This parameter is identical to the fStyle parameter of the REBARBANDINFO structure.

cxMinChild

Minimum width of the child window, in pixels. This parameter is identical to the cxMinChild parameter of the REBARBANDINFO structure.

cyMinChild

Minimum height of the child window, in pixels. This parameter is identical to the cyMinChild parameter of the REBARBANDINFO structure.

cx

Length of the band, in pixels. This parameter is identical to the cx parameter of the REBARBANDINFO structure.

cxIdeal

Ideal width of the band, in pixels. This parameter is identical to the cxIdeal parameter of the REBARBANDINFO structure.

nBand

Zero-based index of the location where the band will be inserted. If you set this parameter to -1, the rebar control will add the new band at the last location.

wPlugInCmdID

The command ID of the plug-in.

Version

Supported on Version 7.00 or later.

Send feedback on this topic to Emurasoft

Messages to Plug-ins

EP_GET_BITMAP	Retrieves bitmap resource IDs for various sizes and color depths for the plug-in displayed on a toolbar.
EP_GET_INFO	Retrieves information about the plug-in.
EP_GET_MASK	Retrieves mask color for the plug-in button on a toolbar.
EP_GET_NAME	Retrieves the plug-in name.
EP_GET_VERSION	Retrieves the plug-in version.
EP_QUERY_PROPERTIES	Queries whether the property is enabled.
EP_QUERY_UNINSTALL	Queries whether the plug-in can be uninstalled.
EP_SET_PROPERTIES	Requests the plug-in to display the properties.
EP_SET_UNINSTALL	Uninstalls the plug-in.
EP_PRE_TRANSLATE_MSG	Called before each Windows message is translated.

These messages are used by Functions to Export PlugInProc.

These constants are defined at the header file (plugin.h).

Send feedback on this topic to Emurasoft

EP_GET_BITMAP

Retrieves bitmap resource IDs for various sizes and color depths for the plug-in displayed on a toolbar.

```
EP_GET_BITMAP
  hwnd = hwndParent;
  wParam = flags;
  lParam = 0;
```

Parameters

hwndParent

The window handle of the EmEditor frame window.

flags

Specifies the bitmap size and color depth for a bitmap to retrieve. It is a combination of the following flags.

Value	Meaning
BITMAP_SMALL	Small bitmap (16x16 pixels)
BITMAP_LARGE	Large bitmap (24x24 pixels)
BITMAP_16_COLOR	16 color bitmap
BITMAP_24BIT_COLOR	24bit color (true color) bitmap
BITMAP_256_COLOR	256 color bitmap
BITMAP_DEFAULT	Default state bitmap
BITMAP_DISABLED	Disabled state bitmap
BITMAP_HOT	Hot state bitmap

Return Values

You must return a bitmap resource ID for the requested size and color depth. If the return value is zero, EmEditor will use GetBitmapID export function to retrieve the small 16-color bitmap.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages to Plug-ins

EP_GET_INFO

Retrieves information about the plug-in.

```
EP_GET_INFO
  hwnd = hwndParent;
  wParam = flag;
  lParam = 0;
```

Parameters

hwndParent

The window handle of the EmEditor frame window.

flag

Specifies the type of information requested. It is one of the following values.

Value	Meaning
EPGI_ALLOW_OPEN_SAME_GROUP	Returns TRUE if the plug-in allows EmEditor to open a new file in the same group.
EPGI_ALLOW_MULTIPLE_INSTANCES	Returns TRUE if the plug-in supports multiple instances. If the plug-in should be allowed to run in more than one frame simultaneously, this message should return TRUE. Note that global variables will be shared when multiple instances are running.
EPGI_MAX_EE_VERSION	Returns the newest version number of supported EmEditor * 1000.
EPGI_MIN_EE_VERSION	Returns the oldest version number of supported EmEditor * 1000.
EPGI_SUPPORT_EE_PRO	Returns TRUE if EmEditor Professional is supported.
EPGI_SUPPORT_EE_STD	Returns TRUE if EmEditor Standard is supported.

Return Values

The return value depends on the flag parameter.

Version

Supported on EmEditor Professional Version 5.00 or later.

Send feedback on this topic to Emurasoft

EP_GET_MASK

Retrieves mask color for the plug-in button on a toolbar.

EP_GET_MASK
hwnd = hwndParent;
wParam = flags;
lParam = 0;

Parameters

hwndParent

The window handle of the EmEditor frame window.

flags

Specifies the bitmap color depth for a bitmap to retrieve.

Value	Meaning
BITMAP_24BIT_COLOR	24bit color (true color) bitmap
BITMAP_256_COLOR	256 color bitmap

Return Values

You must return a mask color for the plug-in button on a toolbar as an RGB(r, g, b) value. A mask color on the bitmap will be replaced with the background color of a toolbar. A mask color for a large bitmap must be the same as a mask color for a small bitmap. Currently, you cannot specify a mask color for 16 color bitmaps. If the return value is zero, EmEditor will use RGB (192,192,192) as a default mask color.

Send feedback on this topic to Emurasoft

EP_GET_NAME

Retrieves the plug-in name

EP_GET_NAME
hwnd = hwndParent;
wParam = cb;
lParam = szName;

Parameters

hwndParent

The window handle of the Plug-ins Settings dialog box.

cb

Specifies the maximum number of characters to copy to the buffer, including the NULL character.

szName

Pointer to the buffer that will receive the text.

Return Values

The return value is the required size for a buffer that can receive the text.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages to Plug-ins

EP_GET_VERSION

Retrieves the plug-in version.

```
EP_GET_VERSION
  hwnd = hwndParent;
   wParam = cb;
   lParam = szVersion;
```

Parameters

hwndParent

The window handle of the Plug-ins Settings dialog box.

cb

Specifies the maximum number of characters to copy to the buffer, including the NULL character.

szVersion

Pointer to the buffer that will receive the text.

Return Values

The return value is the required size for a buffer that can receive the text.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages to Plug-ins

EP_QUERY_PROPERTIES

Queries whether the properties are available.

```
EP_QUERY_PROPERTIES
  hwnd = hwndParent;
   wParam = 0;
   lParam = 0;
```

Parameters

hwndParent

The window handle of the Plug-ins Settings dialog box.

Return Values

You must return TRUE if the properties are available, or FALSE if the properties are not available.

Send feedback on this topic to Emurasoft

EmEditor Home - EmEditor Help - Plug-in Reference - Messages to Plug-ins

EP_QUERY_UNINSTALL

Queries whether the plug-in can be uninstalled.

```
EP_QUERY_UNINSTALL
  hwnd = hwndParent;
   wParam = 0;
   lParam = 0;
```

Parameters

hwndParent

The window handle of the Plug-ins Settings dialog box.

Return Values

If the plug-in can be uninstalled, the return value is TRUE. If the plug-in cannot be uninstalled, the return value is FALSE.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages to Plug-ins

EP_SET_PROPERTIES

Requests the plug-in to display the properties.

```
EP_SET_PROPERTIES
  hwnd = hwndParent;
   wParam = 0;
   lParam = 0;
```

Parameters

hwndParent

The window handle of the Plug-ins Settings dialog box.

Return Values

You must return TRUE if the properties have been displayed, or FALSE if the properties have not been displayed.

EmEditor Home - EmEditor Help - Plug-in Reference - Messages to Plug-ins

EP_SET_UNINSTALL

Uninstalls the plug-in.

```
EP_SET_UNINSTALL
  hwnd = hwndParent;
   wParam = (WPARAM) (LPTSTR) pszUninstallCommand;
   lParam = (LPARAM) (LPTSTR) pszUninstallParam;
```

Parameters

hwndParent

The window handle of the Plug-ins Settings dialog box.

pszUninstallCommand

Specifies the command to run when the return value is UNINSTALL_RUN_COMMAND.

pszUninstallParam

Specifies the parameter for the command to run when the return value is UNINSTALL_RUN_COMMAND.

Return Values

The return value is one of following values.

UNINSTALL_FALSE	does not uninstall.
UNINSTALL_SIMPLE_DELETE	simply removes the plug-in DLL file.
UNINSTALL_RUN_COMMAND	runs the specified command.

If the return value is TRUE, the plug-in will be uninstalled. If the return value is FALSE, the plug-in will not be uninstalled.

Version

Supported on Version 3.00 or later. However, the return value UNINSTALL_RUN_COMMAND and pszUninstallCommand parameter and pszUninstallParam parameter are supported on Version 6.00 or later.

Send feedback on this topic to Emurasoft

EP_PRE_TRANSLATE_MSG

Called before each Windows message is translated.

```
EP_PRE_TRANSLATE_MSG
  hwnd = hwndView;
  wParam = 0;
  lParam = (LPARAM) (MSG*) pMsg;
```

Parameters

hwndView

The window handle to the EmEditor view.

pMsg

The pointer to the window message before translated.

Return Values

If the return value is TRUE, the message is not be continued to be translated or dispatched. If the return value is FALSE, the message is continued to be translated or dispatched.

Version

Supported on Version 6.00 or later.

Send feedback on this topic to Emurasoft

// EmEditor Plug-In definition file // v3.08 (6 Jan 2001) CCustomizeInfo::m_bIgnoreColorPrint, CCustomizeInfo::m_bNoFullPathIfNotActive added. // v3.12 (16 Jan 2001) EEID_ALL_PROP added. // CCustomizeInfo::m_abUrlChar is now BYTE, not bool. =2 means not permitted at end of URL. // v3.13 (15 Mar 2001) EEID_NEW_PASTE_PREFIX, EEID_NEW_PASTE_PREFIX_RETURN, EEID_CUSTOMIZE_TRAY, // EEID_INSERT_CR_LF, EEID_MRU_FONT1 added. // CCustomizeInfo::m_bSaveOverwrite, CCustomizeInfo::m_bNoChangeCrLf, // CCustomizeInfo::m_bShowSpace, CCustomizeInfo::m_bWordWrapMark added. // EI_IS_PROPORTIONAL_FONT added. // v3.14 (23 Apr 2001) CCustomizeInfo::m_bPrintSeparatingLines, CCustomizeInfo::m_bSameFontPrint, // CCustomizeInfo::m_bHiliteCorresParen added. // EEID_DELETE_RIGHT_WORD, EEID_NEXT_PAREN, EEID_SHIFT_NEXT_PAREN added. // CHECK_FILE_CHANGED_EXCLUSIVE added. // v3.15 (22 May 2001) CCustomizeInfo::m_bDetectUTF8, CCustomizeInfo::m_bDetectCharset, // CCustomizeInfo::m_bDetectAll, CCustomizeInfo::m_bDeleteSpaceEnd, // CCustomizeInfo::m_bSaveAsEntity, CCustomizeInfo::m_bShowControl added. // Changed LOAD_FILE_INFO structure. // CHARSET_OEM added. // EEID_CHARSET_OEM, EEID_TRIM_RIGHT added. // CODEPAGE_DETECT_UTF8, CODEPAGE_DETECT_CHARSET, CODEPAGE_DETECT_ALL added. // v3.16 (21 Jun 2001) EEID_FILE_RELOAD_DETECT_ALL added. // SMART_COLOR_ constants redefined. // CColorInfo class redefined. // CCustomizeInfo::m_nHiliteTag, m_nHiliteMultiComment removed. // CCustomizeInfo::m_nSpecialSyntax, m_chEscape, m_bPasteAnsi, m_bQuoteType, // m_szScriptBegin, m_szScriptEnd, m_szLineComment1, m_szLineComment2 added. // v3.17 (25 Jul 2001) EEID_DELETE_LEFT_WORD, EEID_FILE_NEW_CONFIG, // EEID_NEW_CONFIG_POPUP, EEID_FONT_POPUP added. // CCustomizeInfo::m_bSaveAsNameEntity, m_chIndentBegin, m_bNewTemplate, // m_chIndentEnd, m_chEndOfStatement added. // v3.18 (29 Aug 2001) EEID_RELOAD_POPUP, EEID_DELETE_PANES added. // MAX_RECENT_FILE defined. // v3.19 (1 Oct 2001) EVENT_CHAR added. // EEID_SHOW_PLUGINS_BAR added. // SIGNATURE_PIB_LIST is superseded by SIGNATURE_PIB_LIST_2. // v3.20 (21 Nov 2001) EP_GET_BITMAP, EP_GET_MASK added. // v3.22 (2 Jan 2002) EEID_PRINT_PREVIEW, EEID_WINDOW_TOP, EEID_WINDOW_BOTTOM, // EEID_WINDOW_RIGHT, EEID_WINDOW_LEFT, EEID_HOME_TEXT, EEID_SHIFT_HOME_TEXT, // EEID_KEYBOARD_MAP added. // v3.23 (31 Jan 2002) EEID_WINDOW_SPLIT_HORZ, EEID_WINDOW_SPLIT_VERT // EEID_CONTEXT_MENU, EEID_DELETE_LEFT_LINE, // EEID_INSERT_GRAVE, EEID_INSERT_ACUTE, EEID_INSERT_CIRCUMFLEX, EEID_INSERT_TILDE, // EEID_INSERT_DIAERESIS, EEID_INSERT_RING_ABOVE, EEID_INSERT_LIGATURE, EEID_INSERT_CEDILLA, // EEID_INSERT_STROKE, EEID_INSERT_INVERTED_QUESTION, EEID_INSERT_INVERTED_EXCLAMATION, // EEID_INSERT_COPYRIGHT, EEID_INSERT_REGISTERED, EEID_INSERT_TRADEMARK, EEID_INSERT_EURO added. // v3.24 (10 Feb 2002) EVENT_FILE_OPENED event added. EEID_WRAP_BY_PAPER added. // v3.28 (1 Oct 2002) FLAG_FIND_REG_EXP and FLAG_FIND_CLOSE added. // v3.29 (6 Nov 2002) EEID_SHOW_TOOLS_BAR, EEID_BOOKMARK_TOGGLE, EEID_BOOKMARK_NEXT, EEID_BOOKMARK_PREV, // EEID_BOOKMARK_CLEAR, EEID_CUSTOMIZE_TOOLS, EEID_TOOL1 added. // v3.31 (17 Dec 2002) EEID_RETRIEVE_FIND_TEXT, EEID_COPY_FILE_PATH, EEID_COPY_FILE_DIR, EEID_DUPLICATE_LINE, // EEID_LOAD_WORKSPACE, EEID_SAVE_WORKSPACE, EEID_SAVE_WORKSPACE_EXIT_ALL, EEID_SAVE_WORKSPACE_QUIT_ALL, //

EEID_LOGICAL_HOME_TEXT, EEID_SHIFT_LOGICAL_HOME_TEXT, EEID_WINDOW_SPLIT_HORZ_FIX, EEID_WINDOW_SPLIT_VERT_FIX, // EEID_SHOW_WINDOWS_BAR added. // CCustomizeInfo::m_bShowScrollOnlyActive, CCustomizeInfo::m_bWrapPagePrint added. // EI_GET_NEXT_BOOKMARK added. // v3.32 (25 Jan 2003) CCustomizeInfo::m_nMaxFindHilite added. CColorInfo structure removed. // // v4.00 (18 Dec 2003) EE_SET_SEL_TYPE, EE_GET_STATUSA, EE_GET_STATUSW, EE_INSERT_FILEA, EE_INSERT_FILEW added. // EE_INSERT_STRINGA, EE_INSERT_STRINGW extended to use lParam. // EE_GET_VERSION extended to use wParam. // Editor_GetVersionEx, VERSION_PRO, VERSION_STD added. // EI_GET_HILITE_FIND, EI_SET_HILITE_FIND, EI_GET_APP_VERSIONA, EI_GET_APP_VERSIONW, EI_GET_READ_ONLY, // EI_IS_WINDOW_COMBINED, EI_WINDOW_COMBINE added. // EEID_SHOW_BAR_TITLE, EEID_LOCK_TOOLBARS, EEID_WINDOW_COMBINE, EEID_WINDOW_ALWAYS_TOP_ON, EEID_WINDOW_ALWAYS_TOP_OFF, EEID_MOVE_LAST_EDIT, // EEID_MACRO_SAVE, EEID_MACRO_SELECT, EEID_MACRO_EDIT, EEID_MACRO_SELECT_THIS, EEID_CUSTOMIZE_MACRO, EEID_BOOKMARK_NEXT_WITHIN, // EEID_BOOKMARK_PREV_WITHIN, EEID_BOOKMARK_SET, EEID_BOOKMARK_RESET, EEID_SPACE_TO_TAB, EEID_TABIFY, EEID_UNTABIFY, EEID_INDENT, // EEID_UNINDENT, EEID_MACRO_HELP, EEID_MACRO_HELP_WORD, EEID_REPLACE_IN_FILES, EEID_QUIT_ALL, EEID_MACRO_RUN_OPTIONS, EEID_INSERT_CARON, // EEID_VIEW_MARKS, EEID_EDIT_COMMENT, EEID_EDIT_UNCOMMENT, EEID_INCREASE_FONT_SIZE, EEID_DECREASE_FONT_SIZE added. // v4.01 (27 Dec 2003) EI_IS_UNDO_COMBINED added. // EEID_FIND_NEXT_UNICODE, EEID_FIND_PREV_UNICODE, EEID_ERASE_UNICODE_HILITE added. // v4.02 (30 Jan 2004) EE_FIND_IN_FILES, EE_REPLACE_IN_FILES added. // v4.03 (25 Feb 2004) EE_GET_ANCHOR_POS, EE_SET_ANCHOR_POS added. EE_SET_CARET_POS extended. // v4.05 (30 Apr 2004) "cb" field of GET_LINE_INFO was changed to "cch" for clarity. // v4.10 (1 Jan 2005) EEID_JOIN_LINES, EEID_SPLIT_LINES, EEID_IMPORT_EXPORT, EEID_CAPITALIZE, EEID_WINDOW_MOVE_NEXT, EEID_WINDOW_MOVE_PREV, EEID_CLOSE_ALL_OTHERS added. // FLAG_CAPITALIZE flag added for EE_CONVERT message. // v4.13 (15 Feb 2005) EEID_WINDOW_SPLIT_HORZ_TOGGLE, EEID_WINDOW_SPLIT_VERT_TOGGLE added. // v5.00 (24 Nov 2005) EE_GET_REDRAW added. // EVENT_CREATE_FRAME, EVENT_CLOSE_FRMAE, EVENT_DOC_SEL_CHANGED, EVENT_DOC_CLOSE events added. // EEID_GROUP_CLOSE_ALL, EEID_GROUP_CLOSE_OTHERS, EEID_GROUP_CLOSE_LEFT, EEID_GROUP_CLOSE_RIGHT, EEID_NEW_GROUP, EEID_NEW_GROUP_MINIMIZE, // EEID_NEW_GROUP_CASCADE, EEID_NEW_GROUP_HORZ, EEID_NEW_GROUP_VERT, EEID_MOVE_PREV_GROUP, EEID_MOVE_NEXT_GROUP, // EEID_SORT_FILE_NAME, EEID_SORT_TYPE, EEID_SORT_MODIFIED, EEID_SORT_ZORDER, EEID_SORT_ASCENDING, EEID_SORT_DESCENDING, // EEID_AUTO_SORT, EEID_RESTORE_POS, // EEID_CUSTOMIZE_FILE, EEID_CUSTOMIZE_SEARCH, EEID_CUSTOMIZE_HISTORY, EEID_CUSTOMIZE_WINDOW, EEID_CUSTOMIZE_TAB, // EEID_CUSTOMIZE_STATUS, EEID_CUSTOMIZE_ADVANCED, EEID_WINDOW_COMBINE_ON, EEID_WINDOW_COMBINE_OFF added. // EE_SAVE_FILEA, EE_SAVE_FILEW, EE_GET_MODIFIED, EE_GET_CONFIGA, EE_GET_CONFIGW, EE_SET_CONFIGA, EE_SET_CONFIGW, EE_INFO extended to use HIWORD(wParam) = iDoc. // Editor_DocSaveFileA, Editor_DocSaveFileW, Editor_DocGetModified, Editor_DocGetConfigA, Editor_DocGetConfigW // Editor_DocSetConfigA, Editor_DocSetConfigW, Editor_DocInfo macros added. // EI_GET_DOC_COUNT, EI_INDEX_TO_DOC, EI_DOC_TO_INDEX, EI_ZORDER_TO_DOC, EI_DOC_TO_ZORDER, EI_GET_ACTIVE_INDEX, EI_SET_ACTIVE_INDEX, // EI_GET_FULL_TITLEA, EI_GET_FULL_TITLEW, EI_GET_SHORT_TITLEA, EI_GET_SHORT_TITLEW, EI_GET_SAVE_AS_TITLEA, EI_GET_SAVE_AS_TITLEW, // EI_MOVE_ORDER, EI_CLOSE_DOC, EI_SAVE_DOC added for EE_INFO commands. // HEEDOC (handle to document) type defined. // Type changes for x64 (EE_GET_SEL_TEXTA, EE_GET_SEL_TEXTW, EE_GET_LINES, EE_GET_LINEA, EE_GET_LINEW, EE_GET_CARET_POS // EE_DEV_TO_VIEW, EE_GET_PAGE_SIZE, EE_GET_SCROLL_POS, EE_LINE_FROM_CHAR, EE_LINE_INDEX, EE_LOGICAL_TO_SERIAL, EE_LOGICAL_TO_VIEW, // EE_SERIAL_TO_LOGICAL, EE_SET_CARET_POS, EE_SET_SCROLL_POS, EE_VIEW_TO_DEV, EE_VIEW_TO_LOGICAL, EE_GET_SEL_START, // EE_GET_SEL_END, EE_SET_SEL_LENGTH, EE_SET_SEL_VIEW, EE_GET_MARGIN, EE_GET_STATUSA, EE_GET_STATUSW, EE_GET_ANCHOR_POS, // EE_SET_ANCHOR_POS messages, // GET_LINE_INFO, GREP_INFOW, GREP_INFOA, LOAD_FILE_INFO, LOAD_FILE_INFO_EX structures // EE_DO_IDLE message (Editor_DoIdle) added. // EE_GET_SEL_TYPE, EE_SET_SEL_TYPE, EE_SET_SCROLL_POS expanded to use wParam. // EP_GET_INFO added for plugin message. // v6.00 (7 Jun 2006) EEID_ACTIVE_PANE, EEID_OUTLINE_COLLAPSE_ALL, EEID_OUTLINE_EXPAND_ALL, EEID_OUTLINE_TOGGLE_LINE, // EEID_OUTLINE_COLLAPSE_LINE, EEID_OUTLINE_EXPAND_LINE, EEID_OUTLINE_NEXT_NODE, EEID_OUTLINE_PREV_NODE, // EEID_SHIFT_NEXT_NODE, EEID_SHIFT_PREV_NODE, EEID_RESTORE_DELETED, EEID_VIEW_OUTPUT command added. // EE_CUSTOM_BAR_OPEN, EE_CUSTOM_BAR_CLOSE, EE_MATCH_REGEX, EE_FIND_REGEX, EE_GET_OUTLINE_LEVEL, EE_SET_OUTLINE_LEVEL, // EE_SHOW_OUTLINE, EE_ENUM_CONFIG messages added. // EVENT_IDLE, EVENT_CUSTOM_BAR_CLOSED, EVENT_CUSTOM_BAR_CLOSING events added. // EP_PRE_TRANSLATE_MSG added for plugin message. // // v7.00 (18 Dec 2007) m_nKanjiRead renamed to m_nEncodingRead, m_nEncodeNew renamed to m_nEncodingNew, m_nEncodeWrite renamed to m_nEncodingWrite // EE_TOOLBAR_OPEN, EE_TOOLBAR_CLOSE, EE_TOOLBAR_SHOW, EE_HELP, EE_REG_SET_VALUE, EE_REG_QUERY_VALUE, EE_QUERY_STRING, // EE_KEYBOARD_PROP, EE_GET_ACCEL_ARRAY, EE_OUTPUT_STRING, EE_OUTPUT_DIR, EE_ENUM_HIGHLIGHT messages added. // Editor_ToolbarOpen, Editor_ToolbarClose, Editor_ToolbarShow, Editor_Help, Editor_RegSetValue, Editor_RegQueryValue, Editor_QueryString, // Editor_KeyboardProp, Editor_GetAccelArray, Editor_OutputString, Editor_OutputDir, Editor_EnumHighlight inline functions added. // TOOLBAR_INFO, REG_SET_VALUE_INFO, REG_QUERY_VALUE_INFO structures added. // EVENT_TOOLBAR_CLOSED, EVENT_TOOLBAR_SHOW events added. // m_bVirtualSpace member added to CCustomizeInfo. // EI_GET_CURRENT_FOLDER, EI_IS_LARGE_DOC flag added. // EE_GET_LINES, EE_GETLINEW, EE_GETLINEA (GET_LINE_INFO structure) supports iDoc parameter. // Editor_DocGetLines inline functions added. // #pragma once #ifdef __cplusplus #define EE_STRICT // uses inline functions instead of macros #endif #ifndef CLR_NONE #define CLR_NONE 0xFFFFFFFFL #endif #define REG_VERSION 3 #define MAX_HIGHLIGHT_COLOR 10 #define RETURN_METHOD_BOTH 0 #define RETURN_METHOD_CR_ONLY 1 #define RETURN_METHOD_LF_ONLY 2 #define WRAP_NONE 0 #define WRAP_BY_CHAR 1 #define WRAP_BY_WINDOW 2 #define WRAP_BY_PAPER 3 #define MAX_WRAP_MODE 4 #define MIN_MARGIN 16 #define MAX_MARGIN 0x7fff // inclusive #define MIN_LINE_SPACE 0 #define MAX_LINE_SPACE 30 // inclusive #define MIN_CHAR_SPACE 0 #define MAX_CHAR_SPACE 30 // inclusive #define MIN_FIND_HILITE 0 #define MAX_FIND_HILITE 30 // inclusive #define SPECIAL_SYNTAX_NONE 0 #define SPECIAL_SYNTAX_HTML 1 #define SPECIAL_SYNTAX_HTML_EMBEDDED 2 #define MAX_SPECIAL_SYNTAX 3 #define MAX_FIND_HISTORY 32 #define MIN_RECENT_FILE 0 #define MAX_RECENT_FILE 64 #define DEF_RECENT_FILE 8 #define DEFAULT_COLOR (ULONG_MAX-1) #define TRANSPARENT_COLOR (ULONG_MAX-2) #define SIGNATURE_FACE_LIST 0x00FF0000 #define SIGNATURE_HILITE_LIST 0x00FF0100 #define SIGNATURE_FIND_LIST 0x00FF0200 #define SIGNATURE_PIK_LIST 0x00FF0300 #define SIGNATURE_PIB_LIST_2 0x00FF0401 #define SIGNATURE_ASSOCIATE_LIST 0x00FF0500 #define SIGNATURE_CODEPAGE_LIST_2 0x00FF0601 #define SIGNATURE_MENU_LIST 0x00FF0700 #define SIGNATURE_MENU_LIST_2 0x00FF0701 #define SIGNATURE_TOOL_LIST 0x00FF0800 #define SIGNATURE_TOOL_LIST_2 0x00FF0801 #define SIGNATURE_PIK_T_LIST 0x00FF0900 #define SIGNATURE_WORKSPACE_LISTW 0x00FF0A00 #define SIGNATURE_WORKSPACE_LISTA 0x00FF0A01 #define SIGNATURE_WORKSPACE_LISTW_2 0x00FF0A02 #define SIGNATURE_WORKSPACE_LISTA_2 0x00FF0A03 #define SIGNATURE_WORKSPACE_LISTW_3 0x00FF0A06 #define SIGNATURE_WORKSPACE_LISTA_3 0x00FF0A07 #define SIGNATURE_PIK_M_LIST 0x00FF0B00 #define SIGNATURE_MACRO_LANG_LIST 0x00FF0C00 #define SIGNATURE_THEME_LIST 0x00FF0D00 #define MAX_CODEPAGE_NAME 80 #define CHARSET_DEFAULT 0 #define CHARSET_ARABIC 1 #define CHARSET_BALTIC 2 #define CHARSET_CENTRAL_EUROPE 3 #define CHARSET_CHINESE_SIMPLIFIED 4 #define CHARSET_CHINESE_TRADITIONAL 5 #define CHARSET_CYRILLIC 6 #define CHARSET_GREEK 7 #define CHARSET_HEBREW 8 #define CHARSET_JAPANESE 9 #define CHARSET_KOREAN 10 #define CHARSET_THAI 11 #define CHARSET_TURKISH 12 #define CHARSET_VIETNAMESE 13 #define CHARSET_WESTERN_EUROPE 14 #define CHARSET_OEM 15 #define CHARSET_RESERVED_4 16 #define CHARSET_RESERVED_3 17 #define CHARSET_RESERVED_2

18 #define CHARSET_RESERVED_1 19 #define MAX_CHARSET 20 #define MAX_USED_CHARSET 16 #define CODEPAGE_ANSI 65536 #define CODEPAGE_UNICODE 65537 #define CODEPAGE_UTF16LE CODEPAGE_UNICODE #define CODEPAGE_UNICODE_BIGENDIAN 65538 #define CODEPAGE_UTF16BE CODEPAGE_UNICODE_BIGENDIAN #define CODEPAGE_UTF8 65001 #define CODEPAGE_UTF7 65000 #define CODEPAGE_ANSI_FIRST 4 #define CODEPAGE_ANSI_LAST 64999 #define CODEPAGE_932 932 // Japanese Shift-JIS #define CODEPAGE_JIS 65616 // obsolete, Japanese JIS, use 50220 instead #define CODEPAGE_EUC 65617 // obsolete, Japanese EUC, use 51932 instead. #define CODEPAGE_DETECT_UNICODE 0x00020000 #define CODEPAGE_DETECT_UTF8 0x00040000 // v3.15 #define CODEPAGE_DETECT_CHARSET 0x00080000 // v3.15 #define CODEPAGE_DETECT_ALL 0x00100000 // v3.15 #define CODEPAGE_MASK 0x0001ffff #define CODEPAGE_AUTO_SJIS_JIS 66049 // obsolete #define CODEPAGE_AUTO_SJIS_JIS_EUC 66050 // obsolete, use 50932 instead. #define CODEPAGE_UNKNOWN 66304 // internal use only #define CODEPAGE_MAYBE_EUC 66305 // internal use only #define CODEPAGE_CONFIG 66307 // internal use only #define DEF_UNDO_BUFFER_SIZE 1000000 #define MIN_UNDO_BUFFER_SIZE 10 #define MAX_UNDO_BUFFER_SIZE 0x0aaaaaa9 #define MAX_PLUG_IN_NAME 80 #define MAX_FILTER_LENGTH 256 #define MAX_CONFIG_NAME 260 #define MAX_ASSOCIATE_LENGTH 16 #define MAX_THEME_NAME 32 #define MAX_HEADER 116 // was 128 before v3.16 #define MAX_FOOTER 116 // was 128 before v3.16 #define MAX_KINSOKU_BEGIN 128 #define MAX_KINSOKU_END 128 #define MAX_MULTI_COMMENT_BEGIN 16 #define MAX_MULTI_COMMENT_END 16 #define MAX_LINE_COMMENT 4 // v3.16 #define MAX_SCRIPT_BEGIN 8 // v3.16 #define MAX_SCRIPT_END 8 // v3.16 #define MAX_PREFIX_LENGTH 80 #define MAX_FILE_FILTER 128 #define MAX_DEF_EXT 128 #define MAX_PREFIX_LIST 40 #define SMART_COLOR_NONE ((BYTE)(-1)) #define SMART_COLOR_NORMAL 0 #define SMART_COLOR_SEL 1 #define SMART_COLOR_CURLINE 2 #define SMART_COLOR_QUOTE 3 #define SMART_COLOR_FIND 4 #define SMART_COLOR_LINK_URL 5 #define SMART_COLOR_LINK_ID 6 #define SMART_COLOR_LINK_TAG 7 #define SMART_COLOR_SINGLE_QUOTES 8 #define SMART_COLOR_DOUBLE_QUOTES 9 #define SMART_COLOR_COMMENT 10 #define SMART_COLOR_SCRIPT 11 #define SMART_COLOR_BRACES 12 #define SMART_COLOR_IN_TAG 13 #define SMART_COLOR_HILITE_1 14 #define SMART_COLOR_HILITE_2 15 #define SMART_COLOR_HILITE_3 16 #define SMART_COLOR_HILITE_4 17 #define SMART_COLOR_HILITE_5 18 #define SMART_COLOR_HILITE_6 19 #define SMART_COLOR_HILITE_7 20 #define SMART_COLOR_HILITE_8 21 #define SMART_COLOR_HILITE_9 22 #define SMART_COLOR_HILITE_10 23 #define SMART_COLOR_RETURN 24 #define SMART_COLOR_LINE 25 #define SMART_COLOR_PAGE_LINE 26 #define SMART_COLOR_LINE_NUMBER 27 #define SMART_COLOR_RULER 28 #define SMART_COLOR_OUTSIDE 29 #define MAX_SMART_COLOR 30 #define MAX_SMART_COLOR_FIND (MAX_SMART_COLOR + (MAX_FIND_HILITE + 1)) // 61 #ifndef UNICODE #define SMART_COLOR_NON_UNICODE 0x40 #define SMART_COLOR_MASK 0x3f #endif #define SMART_COLOR_FONT_NORMAL 0 #define SMART_COLOR_FONT_UNDERLINE 1 #define SMART_COLOR_FONT_BOLD 2 #define SMART_COLOR_FONT_ITALIC 3 #define QUOTE_NONE 0 #define QUOTE_SINGLE 1 #define QUOTE_DOUBLE 2 #define QUOTE_BOTH 3 #define QUOTE_CONTINUE 4 // v6 #define CUSTOM_BAR_LEFT 0 #define CUSTOM_BAR_TOP 1 #define CUSTOM_BAR_RIGHT 2 #define CUSTOM_BAR_BOTTOM 3 #define MAX_CUSTOM_BAR 4 // v7 #define EEREG_COMMON (0x7fffff00) // HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Common or eeCommon.ini\[Common] #define EEREG_REGIST (0x7fffff01) // HKEY_CURRENT_USER\Software\EmSoft\Regist or eeCommon.ini\[Regist] #define EEREG_MACROS (0x7fffff02) // HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Macros or eeCommon.ini\[Macros] #define EEREG_PLUGINS (0x7fffff03) // HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\PlugIns or eeCommon.ini\[PlugIns] #define EEREG_RECENT_FILE_LIST (0x7fffff04) // HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent File List or eeCommon.ini\[Recent File List] #define EEREG_RECENT_FOLDER_LIST (0x7fffff05) // HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Folder List or eeCommon.ini\[Recent Folder List] #define EEREG_RECENT_FONT_LIST (0x7fffff06) // HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Font List or eeCommon.ini\[Recent Font List] #define EEREG_RECENT_INSERT_LIST (0x7fffff07) // HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Recent Insert List or eeCommon.ini\[Recent Insert List] #define EEREG_AUTOSAVE (0x7fffff08) // HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3 \AutoSave or eeCommon.ini\[AutoSave] #define EEREG_LM_COMMON (0x7fffff11) // HKEY_LOCAL_MACHINE\SOFTWARE\EmSoft\EmEditor v3\Common or eeLM.ini\[Common] #define EEREG_LM_REGIST (0x7fffff12) // HKEY_LOCAL_MACHINE\SOFTWARE\EmSoft\Regist or eeLM.ini\[Regist] #define EEREG_CONFIG (0x7fffff20) // HKEY_CURRENT_USER\Software\EmSoft\EmEditor v3\Config\(pszConfig) or eeConfig.ini\[(pszConfig)] #define EEREG_EMEDITORPLUGIN (0x7fffff30) // HKEY_CURRENT_USER\Software\EmSoft\EmEditorPlugIns or eePlugin.ini\[(pszConfig)] #define IS_EEREG_COMMON(x) ((DWORD)x >= EEREG_COMMON && (DWORD)x < EEREG_LM_COMMON) #define IS_EEREG_LM (x) ((DWORD)x >= EEREG_LM_COMMON && (DWORD)x < EEREG_CONFIG) #define EE_REG_VARIABLE_SIZE 1 // events #define EVENT_SEL_CHANGED 0x00000010 #define EVENT_CARET_MOVED 0x00000020 #define EVENT_MODIFIED 0x00000040 #define EVENT_SCROLL 0x00000080 #define EVENT_CONFIG_CHANGED 0x00000100 #define EVENT_CHANGE 0x00000200 #define EVENT_CREATE 0x00000400 #define EVENT_CLOSE 0x00000800 #define EVENT_KILL_FOCUS 0x00001000 #define EVENT_SET_FOCUS 0x00002000 #define EVENT_CHAR 0x00004000 #define EVENT_FILE_OPENED 0x00008000 // new events for v5 #define EVENT_CREATE_FRAME 0x00010000 #define EVENT_CLOSE_FRAME 0x00020000 #define EVENT_DOC_SEL_CHANGED 0x00040000 #define EVENT_TAB_MOVED 0x00080000 #define EVENT_DOC_CLOSE 0x00100000 // new events for v6 #define EVENT_IDLE 0x00200000 #define EVENT_CUSTOM_BAR_CLOSED 0x00400000 #define EVENT_CUSTOM_BAR_CLOSING 0x00800000 // new events for v7 #define EVENT_TOOLBAR_CLOSED 0x01000000 #define EVENT_TOOLBAR_SHOW 0x02000000 typedef void *HEEDOC; // EE_LOAD_FILE typedef struct _LOAD_FILE_INFO { size_t cbSize; // sizeof(LOAD_FILE_INFO) UINT nCP; BOOL bDetectUnicode; BOOL bDetectAll; BOOL bDetectCharset; BOOL bDetectUTF8; } LOAD_FILE_INFO; // EE_LOAD_FILE_EX typedef struct _LOAD_FILE_INFO_EX { size_t cbSize; // sizeof(LOAD_FILE_INFO_EX) UINT nCP; BOOL bDetectUnicode; BOOL bDetectAll; BOOL bDetectCharset; BOOL bDetectUTF8; UINT nFlags; } LOAD_FILE_INFO_EX; #define LFI_ALLOW_NEW_WINDOW 1 typedef struct _GET_LINE_INFO { UINT_PTR cch; UINT flags; UINT_PTR yLine; } GET_LINE_INFO; typedef struct _GREP_INFOW { size_t cbSize; // sizeof(GREP_INFOW) UINT nCP; UINT nFlags; LPCWSTR pszFind; LPCWSTR pszReplace; LPCWSTR pszPath; LPCWSTR pszBackupPath; LPCWSTR pszFilesToIgnore; } GREP_INFOW; typedef struct _GREP_INFOA { size_t cbSize; // sizeof(GREP_INFOA) UINT nCP; UINT nFlags; LPCSTR pszFind; LPCSTR pszReplace; LPCSTR pszPath; LPCSTR pszBackupPath; LPCSTR pszFilesToIgnore; } GREP_INFOA; typedef struct _MATCH_REGEX_INFO { size_t cbSize; // sizeof(MATCH_REGEX_INFO) UINT nFlags; LPCWSTR pszRegex; LPCWSTR pszText; } MATCH_REGEX_INFO; typedef struct _FIND_REGEX_INFO { size_t cbSize; // sizeof (FIND_REGEX_INFO) UINT nFlags; LPCWSTR pszRegex; LPCWSTR pszText; LPCWSTR* ppszStart; LPCWSTR* ppszEnd; LPCWSTR* ppszNext; } FIND_REGEX_INFO; typedef struct _CUSTOM_BAR_INFO { size_t cbSize; HWND hwndCustomBar; HWND hwndClient; LPCTSTR pszTitle; int iPos; } CUSTOM_BAR_INFO; #define CLOSED_FRAME_WINDOW 1 #define CLOSED_ANOTHER_CUSTOM_BAR 2 typedef struct _CUSTOM_BAR_CLOSE_INFO { UINT nID; int iPos; DWORD dwFlags; } CUSTOM_BAR_CLOSE_INFO; #if (defined(_WIN64) || defined(_W64)) typedef struct tagPOINT_PTR { LONG_PTR x; LONG_PTR y; } POINT_PTR, *PPOINT_PTR; typedef struct tagSIZE_PTR { LONG_PTR cx; LONG_PTR cy; } SIZE_PTR, *PSIZE_PTR; #else typedef struct tagPOINT_PTR { LONG x; LONG y; } POINT_PTR, *PPOINT_PTR; typedef struct tagSIZE_PTR { LONG cx; LONG cy; } SIZE_PTR, *PSIZE_PTR; #endif #define HISTORY_INSERT_CHAR 0 #define HISTORY_BACK_SPACE 1 #define HISTORY_DELETE_CHAR 2 #define HISTORY_INSERT_STRING 3 #define HISTORY_DELETE_SMALL_STRING 5 #define HISTORY_INSERT_TAB_SEL 6 #define HISTORY_MODIFIED 0x00010000L #define HISTORY_COMBINED 0x00020000L #define HISTORY_CR_ONLY 0x00040000L #define HISTORY_LF_ONLY 0x00080000L typedef struct _HISTORY_INFO { size_t cbSize; UINT nFlags; POINT_PTR ptTop; POINT_PTR ptBottom; UINT nChar; LPCWSTR pszString; } HISTORY_INFO; #define TIM_REBAR 0x0001 #define TIM_CLIENT 0x0002 #define TIM_TITLE 0x0004 #define TIM_ID 0x0008 #define TIM_FLAGS 0x0010 #define TIM_STYLE 0x0020 #define TIM_MINCHILD 0x0040 #define TIM_CX 0x0080 #define TIM_CXIDEAL 0x0100 #define TIM_BAND 0x0200 #define TIM_PLUG_IN_CMD_ID 0x0400 typedef struct _TOOLBAR_INFO { size_t cbSize; HWND hwndRebar; HWND hwndClient; LPCTSTR

pszTitle; UINT nMask; UINT nID; UINT nFlags; UINT fStyle; UINT cxMinChild; UINT cyMinChild; UINT cx; UINT cxIdeal; UINT nBand; WORD wPlugInCmdID; // UINT cyChild; // UINT cyMaxChild; // UINT cyIntegral; } TOOLBAR_INFO; typedef struct _REG_SET_VALUE_INFO { size_t cbSize; DWORD dwKey; LPCWSTR pszConfig; LPCWSTR pszValue; DWORD dwType; const BYTE* lpData; DWORD cbData; DWORD dwFlags; } REG_SET_VALUE_INFO; typedef struct _REG_QUERY_VALUE_INFO { size_t cbSize; DWORD dwKey; LPCWSTR pszConfig; LPCWSTR pszValue; DWORD dwType; BYTE* lpData; DWORD* lpcbData; DWORD dwFlags; } REG_QUERY_VALUE_INFO; #define EE_FIRST (WM_USER+0x400) #define EE_GET_CMD_ID (EE_FIRST+0) // (HINSTANCE)lParam = hInstance // returns (UINT)nCmdID #ifdef EE_STRICT inline UINT Editor_GetCmdID(HWND hwnd, HINSTANCE hInstance) { return (UINT)SNDMSG(hwnd, EE_GET_CMD_ID, 0, (LPARAM)hInstance); } #else #define Editor_GetCmdID(hwnd, hInstance) \ (UINT) SNDMSG((hwnd), EE_GET_CMD_ID, 0, (LPARAM)(HINSTANCE)(hInstance)) #endif #define EE_QUERY_STATUS (EE_FIRST+1) // (UINT)wParam = nCmdID, (BOOL*)lParam = &bChecked // returns (BOOL)bEnabled #ifdef EE_STRICT inline BOOL Editor_QueryStatus (HWND hwnd, UINT nCmdID, BOOL* pbChecked) { return (BOOL)SNDMSG(hwnd, EE_QUERY_STATUS, (WPARAM)nCmdID, (LPARAM)pbChecked); } #else #define Editor_QueryStatus(hwnd, nCmdID, pbChecked) \ (BOOL)SNDMSG((hwnd), EE_QUERY_STATUS, (WPARAM)(UINT)(nCmdID), (LPARAM)(BOOL*)pbChecked) #endif #define EE_UPDATE_TOOLBAR (EE_FIRST+2) // (UINT)wParam = nCmdID #ifdef EE_STRICT inline void Editor_UpdateToolbar(HWND hwnd, UINT nCmdID) { SNDMSG(hwnd, EE_UPDATE_TOOLBAR, (WPARAM)nCmdID, 0); } #else #define Editor_UpdateToolbar(hwnd, nCmdID) \ (void) SNDMSG((hwnd), EE_UPDATE_TOOLBAR, (WPARAM)(UINT)(nCmdID), (LPARAM)0) #endif #define EE_GET_SEL_TEXTA (EE_FIRST+3) // (UINT_PTR)wParam = nBufferSize, (LPSTR)lParam = szBuffer // returns (UINT_PTR)nRequiredBufferSize #ifdef EE_STRICT inline UINT_PTR Editor_GetSelTextA(HWND hwnd, UINT_PTR nBufferSize, LPSTR szBuffer) { return (UINT_PTR) SNDMSG(hwnd, EE_GET_SEL_TEXTA, (WPARAM)nBufferSize, (LPARAM)szBuffer); } #else #define Editor_GetSelTextA(hwnd, nBufferSize, szBuffer) \ (UINT_PTR)SNDMSG((hwnd), EE_GET_SEL_TEXTA, (WPARAM)(UINT_PTR)(nBufferSize), (LPARAM) (LPSTR)(szBuffer)) #endif #define EE_GET_SEL_TEXTW (EE_FIRST+46) // (UINT_PTR)wParam = nBufferSize, (LPWSTR)lParam = szBuffer // returns (UINT_PTR)nRequiredBufferSize #ifdef EE_STRICT inline UINT_PTR Editor_GetSelTextW(HWND hwnd, UINT_PTR nBufferSize, LPWSTR szBuffer) { return (UINT_PTR)SNDMSG(hwnd, EE_GET_SEL_TEXTW, (WPARAM)nBufferSize, (LPARAM) szBuffer); } #else #define Editor_GetSelTextW(hwnd, nBufferSize, szBuffer) \ (UINT_PTR)SNDMSG((hwnd), EE_GET_SEL_TEXTW, (WPARAM)(UINT_PTR)(nBufferSize), (LPARAM)(LPWSTR)(szBuffer)) #endif #define EE_GET_LINES (EE_FIRST+4) // wParam = MAKEWPARAM(nLogical, iDoc+1) // returns (UINT_PTR)nTotalLines inline UINT_PTR Editor_DocGetLines(HWND hwnd, int iDoc, int nLogical) { return (UINT_PTR)SNDMSG(hwnd, EE_GET_LINES, (WPARAM)MAKEWPARAM(nLogical, iDoc+1), (LPARAM)0); } #ifdef EE_STRICT inline UINT_PTR Editor_GetLines(HWND hwnd, int nLogical) { return (UINT_PTR)SNDMSG(hwnd, EE_GET_LINES, (WPARAM)(int)(nLogical), (LPARAM)0); } #else #define Editor_GetLines(hwnd, nLogical) \ (UINT_PTR)SNDMSG((hwnd), EE_GET_LINES, (WPARAM)(int)(nLogical), (LPARAM)0) #endif #define EE_GET_LINEA (EE_FIRST+5) // (GET_LINE_INFO*)wParam = pGetLineInfo, (LPSTR)lParam = szString // returns (UINT_PTR)nRequiredBufferSize #ifdef EE_STRICT inline UINT_PTR Editor_GetLineA(HWND hwnd, GET_LINE_INFO* pGetLineInfo, LPSTR szString) { return (UINT_PTR)SNDMSG((hwnd), EE_GET_LINEA, (WPARAM)(GET_LINE_INFO*)(pGetLineInfo), (LPARAM)(LPSTR)(szString)); } #else #define Editor_GetLineA(hwnd, pGetLineInfo, szString) \ (UINT_PTR)SNDMSG((hwnd), EE_GET_LINEA, (WPARAM)(GET_LINE_INFO*)(pGetLineInfo), (LPARAM) (LPSTR)(szString)) #endif #define EE_GET_LINEW (EE_FIRST+47) // (GET_LINE_INFO*)wParam = pGetLineInfo, (LPWSTR)lParam = szString // returns (UINT_PTR)nRequiredBufferSize #ifdef EE_STRICT inline UINT_PTR Editor_GetLineW(HWND hwnd, GET_LINE_INFO* pGetLineInfo, LPWSTR szString) { return (UINT_PTR)SNDMSG((hwnd), EE_GET_LINEW, (WPARAM) (GET_LINE_INFO*)(pGetLineInfo), (LPARAM)(LPWSTR)(szString)); } #else #define Editor_GetLineW(hwnd, pGetLineInfo, szString) \ (UINT_PTR)SNDMSG((hwnd), EE_GET_LINEW, (WPARAM)(GET_LINE_INFO*)(pGetLineInfo), (LPARAM)(LPWSTR)(szString)) #endif #define EE_GET_CARET_POS (EE_FIRST+6) // (int)wParam = nLogical, (POINT_PTR*)lParam = pptPos #ifdef EE_STRICT inline void Editor_GetCaretPos(HWND hwnd, int nLogical, POINT_PTR* pptPos) { SNDMSG(hwnd, EE_GET_CARET_POS, (WPARAM) nLogical, (LPARAM)pptPos); } #else #define Editor_GetCaretPos(hwnd, nLogical, pptPos) \ (void)SNDMSG((hwnd), EE_GET_CARET_POS, (WPARAM)(int)(nLogical), (LPARAM)(POINT_PTR*)(pptPos)) #endif #define EE_DEV_TO_VIEW (EE_FIRST+7) // (POINT_PTR*)wParam = pptDev, (POINT_PTR*)lParam = pptView #ifdef EE_STRICT inline void Editor_DevToView (HWND hwnd, POINT_PTR* pptDev, POINT_PTR* pptView) { SNDMSG((hwnd), EE_DEV_TO_VIEW, (WPARAM)(POINT_PTR*) (pptDev), (LPARAM)(POINT_PTR*)(pptView)); } #else #define Editor_DevToView(hwnd, pptDev, pptView) \ (void)SNDMSG((hwnd), EE_DEV_TO_VIEW, (WPARAM)(POINT_PTR*)(pptDev), (LPARAM)(POINT_PTR*)(pptView)) #endif #define EE_GET_PAGE_SIZE (EE_FIRST+8) // (SIZE_PTR*)lParam = psizePage #ifdef EE_STRICT inline void Editor_GetPageSize(HWND hwnd, SIZE_PTR* psizePage) { SNDMSG((hwnd), EE_GET_PAGE_SIZE, (WPARAM)0, (LPARAM)(SIZE_PTR*)(psizePage)); } #else #define Editor_GetPageSize(hwnd, psizePage) \ (void)SNDMSG((hwnd), EE_GET_PAGE_SIZE, (WPARAM)0, (LPARAM)(SIZE_PTR*)(psizePage)) #endif #define EE_GET_SCROLL_POS (EE_FIRST+9) // (POINT_PTR*)lParam = pptPos #ifdef EE_STRICT inline void Editor_GetScrollPos(HWND hwnd, POINT_PTR* pptPos) { SNDMSG((hwnd), EE_GET_SCROLL_POS, (WPARAM)0, (LPARAM)(POINT_PTR*)(pptPos)); } #else #define Editor_GetScrollPos(hwnd, pptPos) \ (void)SNDMSG((hwnd), EE_GET_SCROLL_POS, (WPARAM)0, (LPARAM)(POINT_PTR*)(pptPos)) #endif #define EE_LINE_FROM_CHAR (EE_FIRST+10) // (int)wParam = nLogical, (UINT_PTR)lParam = nSerialIndex // returns (UINT_PTR)yLine #ifdef EE_STRICT inline UINT_PTR Editor_LineFromChar(HWND hwnd, int nLogical, UINT_PTR nSerialIndex) { return (UINT_PTR)SNDMSG((hwnd), EE_LINE_FROM_CHAR, (WPARAM)(int)(nLogical), (LPARAM)(UINT_PTR)(nSerialIndex)); } #else #define Editor_LineFromChar(hwnd, nLogical, nSerialIndex) \ (UINT_PTR)SNDMSG((hwnd), EE_LINE_FROM_CHAR, (WPARAM) (int)(nLogical), (LPARAM)(UINT_PTR)(nSerialIndex)) #endif #define EE_LINE_INDEX (EE_FIRST+11) // (BOOL)wParam = bLogical, (UINT_PTR)lParam = yLine // returns (UINT_PTR)nSerialIndex #ifdef EE_STRICT inline UINT_PTR Editor_LineIndex(HWND hwnd, BOOL bLogical, UINT_PTR yLine) { return (UINT_PTR)SNDMSG((hwnd), EE_LINE_INDEX, (WPARAM)(BOOL)(bLogical), (LPARAM) (UINT_PTR)(yLine)); } #else #define Editor_LineIndex(hwnd, bLogical, yLine) \ (UINT_PTR)SNDMSG((hwnd), EE_LINE_INDEX, (WPARAM)(BOOL)(bLogical), (LPARAM)(UINT_PTR)(yLine)) #endif #define EE_LOAD_FILEA (EE_FIRST+12) // (LOAD_FILE_INFO_EX*)wParam = plfi // (LPCSTR)lParam = szFileName // returns non-zero if success, 0 if failed #ifdef EE_STRICT inline BOOL Editor_LoadFileA(HWND hwnd, LOAD_FILE_INFO_EX* plfi, LPCSTR szFileName) { return (BOOL)SNDMSG((hwnd), EE_LOAD_FILEA, (WPARAM)plfi, (LPARAM)(LPCSTR)(szFileName)); } #else #define Editor_LoadFileA(hwnd, plfi, szFileName) \ (BOOL)SNDMSG((hwnd), EE_LOAD_FILEA, (WPARAM)(LOAD_FILE_INFO_EX*)plfi, (LPARAM)(LPCSTR)(szFileName)) #endif #define EE_LOAD_FILEW (EE_FIRST+48) // (LOAD_FILE_INFO_EX*)wParam = plfi // (LPCWSTR)lParam = szFileName // returns non-zero if success, 0 if failed #ifdef EE_STRICT inline BOOL Editor_LoadFileW(HWND hwnd, LOAD_FILE_INFO_EX* plfi, LPCWSTR szFileName) { return (BOOL)SNDMSG((hwnd), EE_LOAD_FILEW, (WPARAM)plfi, (LPARAM)(LPCWSTR)(szFileName)); } #else #define Editor_LoadFileW(hwnd, plfi, szFileName) \ (BOOL)SNDMSG((hwnd), EE_LOAD_FILEW, (WPARAM)(LOAD_FILE_INFO_EX*)plfi, (LPARAM)(LPCWSTR)(szFileName)) #endif #define EE_LOGICAL_TO_SERIAL (EE_FIRST+13) // (POINT_PTR*)lParam = pptLogical // returns (UINT_PTR)nSerialIndex #ifdef EE_STRICT inline UINT_PTR Editor_LogicalToSerial(HWND hwnd, POINT_PTR* pptLogical) { return (UINT_PTR)SNDMSG((hwnd), EE_LOGICAL_TO_SERIAL, (WPARAM)0, (LPARAM)(POINT_PTR*)(pptLogical)); } #else #define Editor_LogicalToSerial(hwnd, pptLogical) \ (UINT_PTR)SNDMSG((hwnd), EE_LOGICAL_TO_SERIAL, (WPARAM)0, (LPARAM)(POINT_PTR*)(pptLogical)) #endif #define EE_LOGICAL_TO_VIEW (EE_FIRST+14) // (POINT_PTR*)wParam = pptLogical, (POINT_PTR*)lParam = pptView #ifdef EE_STRICT inline void Editor_LogicalToView(HWND hwnd, POINT_PTR* pptLogical, POINT_PTR* pptView) { SNDMSG((hwnd), EE_LOGICAL_TO_VIEW, (WPARAM)(POINT_PTR*)(pptLogical), (LPARAM) (POINT_PTR*)(pptView)); } #else #define Editor_LogicalToView(hwnd, pptLogical, pptView) \ (void)SNDMSG((hwnd), EE_LOGICAL_TO_VIEW, (WPARAM)(POINT_PTR*)(pptLogical), (LPARAM)(POINT_PTR*)(pptView)) #endif #define EE_SAVE_FILEA (EE_FIRST+16) // (BOOL)wParam = bReplace, (LPSTR)lParam = szFileName // returns non-zero if success, 0 if failed #ifdef EE_STRICT inline BOOL Editor_SaveFileA(HWND hwnd, BOOL bReplace, LPSTR szFileName) { return (BOOL)SNDMSG((hwnd), EE_SAVE_FILEA, (WPARAM)(BOOL)(bReplace), (LPARAM)(LPSTR)(szFileName)); } #else #define Editor_SaveFileA(hwnd, bReplace, szFileName) \ (BOOL)SNDMSG((hwnd), EE_SAVE_FILEA, (WPARAM)(BOOL)(bReplace), (LPARAM)(LPSTR)(szFileName)) #endif

```
#ifdef EE_STRICT inline BOOL Editor_DocSaveFileA( HWND hwnd, int iDoc, BOOL bReplace, LPSTR szFileName ) { return (BOOL)
SNDMSG( (hwnd), EE_SAVE_FILEA, MAKEWPARAM((bReplace), (iDoc)+1), (LPARAM)(LPSTR)(szFileName) ); } #else #define
Editor_DocSaveFileA( hwnd, iDoc, bReplace, szFileName ) \ (BOOL)SNDMSG( (hwnd), EE_SAVE_FILEA, MAKEWPARAM((bReplace),
(iDoc)+1), (LPARAM)(LPSTR)(szFileName) ) #endif #define EE_SAVE_FILEW (EE_FIRST+49) // (BOOL)wParam = bReplace, (LPWSTR)
lParam = szFileName // returns non-zero if success, 0 if failed #ifdef EE_STRICT inline BOOL Editor_SaveFileW( HWND hwnd, BOOL
bReplace, LPWSTR szFileName ) { return (BOOL)SNDMSG( (hwnd), EE_SAVE_FILEW, (WPARAM)(BOOL)(bReplace), (LPARAM)
(LPWSTR)(szFileName) ); } #else #define Editor_SaveFileW( hwnd, bReplace, szFileName ) \ (BOOL)SNDMSG( (hwnd), EE_SAVE_FILEW,
(WPARAM)(BOOL)(bReplace), (LPARAM)(LPWSTR)(szFileName) ) #endif #ifdef EE_STRICT inline BOOL Editor_DocSaveFileW( HWND
hwnd, int iDoc, BOOL bReplace, LPWSTR szFileName ) { return (BOOL)SNDMSG( (hwnd), EE_SAVE_FILEW, MAKEWPARAM
((bReplace), (iDoc)+1), (LPARAM)(LPWSTR)(szFileName) ); } #else #define Editor_DocSaveFileW( hwnd, iDoc, bReplace, szFileName ) \
(BOOL)SNDMSG( (hwnd), EE_SAVE_FILEW, MAKEWPARAM((bReplace), (iDoc)+1), (LPARAM)(LPWSTR)(szFileName) ) #endif #define
EE_SERIAL_TO_LOGICAL (EE_FIRST+17) // (UINT_PTR)wParam = nSerial, (POINT_PTR*)lParam = pptLogical #ifdef EE_STRICT
inline void Editor_SerialToLogical( HWND hwnd, UINT_PTR nSerial, POINT_PTR* pptLogical ) { SNDMSG( (hwnd),
EE_SERIAL_TO_LOGICAL, (WPARAM)(UINT_PTR)(nSerial), (LPARAM)(POINT_PTR*)(pptLogical) ); } #else #define
Editor_SerialToLogical( hwnd, nSerial, pptLogical ) \ (void)SNDMSG( (hwnd), EE_SERIAL_TO_LOGICAL, (WPARAM)(UINT_PTR)
(nSerial), (LPARAM)(POINT_PTR*)(pptLogical) ) #endif #define EE_SET_CARET_POS (EE_FIRST+18) // wParam = MAKEWPARAM
( nLogical, bExtend ) // (POINT_PTR*)lParam = pptPos #ifdef EE_STRICT inline void Editor_SetCaretPos( HWND hwnd, int nLogical,
POINT_PTR* pptPos ) { SNDMSG( (hwnd), EE_SET_CARET_POS, (WPARAM)(int)(nLogical), (LPARAM)(POINT_PTR*)(pptPos) ); } #else
#define Editor_SetCaretPos( hwnd, nLogical, pptPos ) \ (void)SNDMSG( (hwnd), EE_SET_CARET_POS, (WPARAM)(int)(nLogical),
(LPARAM)(POINT_PTR*)(pptPos) ) #endif #ifdef EE_STRICT inline void Editor_SetCaretPosEx( HWND hwnd, int nLogical, POINT_PTR*
pptPos, BOOL bExtend ) { SNDMSG( (hwnd), EE_SET_CARET_POS, MAKEWPARAM(nLogical, bExtend), (LPARAM)(POINT_PTR*)
(pptPos) ); } #else #define Editor_SetCaretPosEx( hwnd, nLogical, pptPos, bExtend ) \ (void)SNDMSG( (hwnd), EE_SET_CARET_POS,
MAKEWPARAM(nLogical, bExtend), (LPARAM)(POINT_PTR*)(pptPos) ) #endif #define EE_SET_SCROLL_POS (EE_FIRST+19) //
(BOOL)wParam = bCanMoveCursor // (POINT_PTR*)lParam = pptPos #ifdef EE_STRICT inline void Editor_SetScrollPos( HWND hwnd,
POINT_PTR* pptPos ) { SNDMSG( (hwnd), EE_SET_SCROLL_POS, (WPARAM)0, (LPARAM)(POINT_PTR*)(pptPos) ); } #else #define
Editor_SetScrollPos( hwnd, pptPos ) \ (void)SNDMSG( (hwnd), EE_SET_SCROLL_POS, (WPARAM)0, (LPARAM)(POINT_PTR*)(pptPos) )
#endif #ifdef EE_STRICT inline void Editor_SetScrollPosEx( HWND hwnd, POINT_PTR* pptPos, BOOL bCanMoveCursor ) { SNDMSG
( (hwnd), EE_SET_SCROLL_POS, (WPARAM)(BOOL)bCanMoveCursor, (LPARAM)(POINT_PTR*)(pptPos) ); } #else #define
Editor_SetScrollPosEx( hwnd, pptPos, bCanMoveCursor ) \ (void)SNDMSG( (hwnd), EE_SET_SCROLL_POS, (WPARAM)(BOOL)
bCanMoveCursor, (LPARAM)(POINT_PTR*)(pptPos) ) #endif #define EE_VIEW_TO_DEV (EE_FIRST+20) // (POINT_PTR*)wParam =
pptView, (POINT_PTR*)lParam = pptDev #ifdef EE_STRICT inline void Editor_ViewToDev( HWND hwnd, POINT_PTR* pptView,
POINT_PTR* pptDev ) { SNDMSG( (hwnd), EE_VIEW_TO_DEV, (WPARAM)(POINT_PTR*)(pptView), (LPARAM)(POINT_PTR*)
(pptDev) ); } #else #define Editor_ViewToDev( hwnd, pptView, pptDev ) \ (void)SNDMSG( (hwnd), EE_VIEW_TO_DEV, (WPARAM)
(POINT_PTR*)(pptView), (LPARAM)(POINT_PTR*)(pptDev) ) #endif #define EE_VIEW_TO_LOGICAL (EE_FIRST+21) // (POINT*)
wParam = pptView, (POINT*)lParam = pptLogical #ifdef EE_STRICT inline void Editor_ViewToLogical( HWND hwnd, POINT_PTR*
pptView, POINT_PTR* pptLogical ) { SNDMSG( (hwnd), EE_VIEW_TO_LOGICAL, (WPARAM)(POINT_PTR*)(pptView), (LPARAM)
(POINT_PTR*)(pptLogical ) ); } #else #define Editor_ViewToLogical( hwnd, pptView, pptLogical ) \ (void)SNDMSG( (hwnd),
EE_VIEW_TO_LOGICAL, (WPARAM)(POINT_PTR*)(pptView), (LPARAM)(POINT_PTR*)(pptLogical ) ) #endif #define
EE_EXEC_COMMAND (EE_FIRST+22) // (UINT)wParam = nCmdID #ifdef EE_STRICT inline BOOL Editor_ExecCommand( HWND
hwnd, UINT nCmdID ) { return (BOOL)SNDMSG( (hwnd), EE_EXEC_COMMAND, (WPARAM)(UINT)(nCmdID), (LPARAM)0 ); } #else
#define Editor_ExecCommand( hwnd, nCmdID ) \ (BOOL)SNDMSG( (hwnd), EE_EXEC_COMMAND, (WPARAM)(UINT)(nCmdID),
(LPARAM)0 ) #endif #define EE_GET_MODIFIED (EE_FIRST+23) // returns (BOOL)bModified #ifdef EE_STRICT inline BOOL
Editor_GetModified( HWND hwnd ) { return (BOOL)SNDMSG( (hwnd), EE_GET_MODIFIED, (WPARAM)0, (LPARAM)0 ); } #else #define
Editor_GetModified( hwnd ) \ (BOOL)SNDMSG( (hwnd), EE_GET_MODIFIED, (WPARAM)0, (LPARAM)0 ) #endif #ifdef EE_STRICT
inline BOOL Editor_DocGetModified( HWND hwnd, int iDoc ) { return (BOOL)SNDMSG( (hwnd), EE_GET_MODIFIED, MAKEWPARAM
(0, (iDoc)+1), (LPARAM)0 ); } #else #define Editor_DocGetModified( hwnd, iDoc ) \ (BOOL)SNDMSG( (hwnd), EE_GET_MODIFIED,
MAKEWPARAM(0, (iDoc)+1), (LPARAM)0 ) #endif #define EE_SET_MODIFIED (EE_FIRST+24) // (BOOL)wParam = bModified #ifdef
EE_STRICT inline void Editor_SetModified( HWND hwnd, BOOL bModified ) { (void)SNDMSG( (hwnd), EE_SET_MODIFIED, (WPARAM)
(BOOL)(bModified), (LPARAM)0 ); } #else #define Editor_SetModified( hwnd, bModified ) \ (void)SNDMSG( (hwnd), EE_SET_MODIFIED,
(WPARAM)(BOOL)(bModified), (LPARAM)0 ) #endif #define EE_GET_SEL_START (EE_FIRST+26) // (int)wParam = nLogical //
(POINT_PTR*)lParam = pptPos #ifdef EE_STRICT inline void Editor_GetSelStart( HWND hwnd, int nLogical, POINT_PTR* pptPos )
{ SNDMSG( (hwnd), EE_GET_SEL_START, (WPARAM)(int)(nLogical), (LPARAM)(POINT_PTR*)(pptPos) ); } #else #define
Editor_GetSelStart( hwnd, nLogical, pptPos ) \ (void)SNDMSG( (hwnd), EE_GET_SEL_START, (WPARAM)(int)(nLogical), (LPARAM)
(POINT_PTR*)(pptPos) ) #endif #define EE_GET_SEL_END (EE_FIRST+27) // (int)wParam = nLogical // (POINT_PTR*)lParam = pptPos
#ifdef EE_STRICT inline void Editor_GetSelEnd( HWND hwnd, int nLogical, POINT_PTR* pptPos ) { SNDMSG( (hwnd),
EE_GET_SEL_END, (WPARAM)(int)(nLogical), (LPARAM)(POINT_PTR*)(pptPos) ); } #else #define Editor_GetSelEnd( hwnd, nLogical,
pptPos ) \ (void)SNDMSG( (hwnd), EE_GET_SEL_END, (WPARAM)(int)(nLogical), (LPARAM)(POINT_PTR*)(pptPos) ) #endif #define
EE_SET_SEL_LENGTH (EE_FIRST+28) // (UINT_PTR)wParam = nLen #ifdef EE_STRICT inline void Editor_SetSelLength( HWND hwnd,
UINT_PTR nLen ) { (void)SNDMSG( (hwnd), EE_SET_SEL_LENGTH, (WPARAM)(UINT_PTR)(nLen), (LPARAM)0 ); } #else #define
Editor_SetSelLength( hwnd, nLen ) \ (void)SNDMSG( (hwnd), EE_SET_SEL_LENGTH, (WPARAM)(UINT_PTR)(nLen), (LPARAM)0 )
#endif #define EE_GET_CONFIGA (EE_FIRST+29) // (LPSTR)lParam = szConfigName #ifdef EE_STRICT inline void Editor_GetConfigA
( HWND hwnd, LPSTR szConfigName ) { SNDMSG( (hwnd), EE_GET_CONFIGA, (WPARAM)0, (LPARAM)(LPSTR)(szConfigName) ); }
#else #define Editor_GetConfigA( hwnd, szConfigName ) \ (void)SNDMSG( (hwnd), EE_GET_CONFIGA, (WPARAM)0, (LPARAM)(LPSTR)
(szConfigName) ) #endif #ifdef EE_STRICT inline void Editor_DocGetConfigA( HWND hwnd, int iDoc, LPSTR szConfigName ) { SNDMSG
( (hwnd), EE_GET_CONFIGA, (WPARAM)MAKEWPARAM(0, (iDoc)+1), (LPARAM)(LPSTR)(szConfigName) ); } #else #define
Editor_DocGetConfigA( hwnd, iDoc, szConfigName ) \ (void)SNDMSG( (hwnd), EE_GET_CONFIGA, (WPARAM)MAKEWPARAM(0, (iDoc)
+1), (LPARAM)(LPSTR)(szConfigName) ) #endif #define EE_GET_CONFIGW (EE_FIRST+50) // (LPWSTR)lParam = szConfigName #ifdef
EE_STRICT inline void Editor_GetConfigW( HWND hwnd, LPWSTR szConfigName ) { SNDMSG( (hwnd), EE_GET_CONFIGW,
(WPARAM)0, (LPARAM)(LPWSTR)(szConfigName) ); } #else #define Editor_GetConfigW( hwnd, szConfigName ) \ (void)SNDMSG( (hwnd),
EE_GET_CONFIGW, (WPARAM)0, (LPARAM)(LPWSTR)(szConfigName) ) #endif #ifdef EE_STRICT inline void Editor_DocGetConfigW
( HWND hwnd, int iDoc, LPWSTR szConfigName ) { SNDMSG( (hwnd), EE_GET_CONFIGW, (WPARAM)MAKEWPARAM(0, (iDoc)+1),
(LPARAM)(LPWSTR)(szConfigName) ); } #else #define Editor_DocGetConfigW( hwnd, iDoc, szConfigName ) \ (void)SNDMSG( (hwnd),
EE_GET_CONFIGW, (WPARAM)MAKEWPARAM(0, (iDoc)+1), (LPARAM)(LPWSTR)(szConfigName) ) #endif #define
EE_SET_CONFIGA (EE_FIRST+30) // (LPCSTR)lParam = szConfigName #ifdef EE_STRICT inline BOOL Editor_SetConfigA( HWND
hwnd, LPSTR szConfigName ) { return (BOOL)SNDMSG( (hwnd), EE_SET_CONFIGA, (WPARAM)0, (LPARAM)(LPSTR)
(szConfigName) ); } #else #define Editor_SetConfigA( hwnd, szConfigName ) \ (BOOL)SNDMSG( (hwnd), EE_SET_CONFIGA, (WPARAM)0,
(LPARAM)(LPSTR)(szConfigName) ) #endif #ifdef EE_STRICT inline BOOL Editor_DocSetConfigA( HWND hwnd, int iDoc, LPSTR
szConfigName ) { return (BOOL)SNDMSG( (hwnd), EE_SET_CONFIGA, (WPARAM)MAKEWPARAM(0, (iDoc)+1), (LPARAM)(LPSTR)
(szConfigName) ); } #else #define Editor_DocSetConfigA( hwnd, iDoc, szConfigName ) \ (BOOL)SNDMSG( (hwnd), EE_SET_CONFIGA,
(WPARAM)MAKEWPARAM(0, (iDoc)+1), (LPARAM)(LPSTR)(szConfigName) ) #endif #define EE_SET_CONFIGW (EE_FIRST+51) //
(LPCWSTR)lParam = szConfigName #ifdef EE_STRICT inline BOOL Editor_SetConfigW( HWND hwnd, LPWSTR szConfigName ) { return
(BOOL)SNDMSG( (hwnd), EE_SET_CONFIGW, (WPARAM)0, (LPARAM)(LPWSTR)(szConfigName) ); } #else #define Editor_SetConfigW
( hwnd, szConfigName ) \ (BOOL)SNDMSG( (hwnd), EE_SET_CONFIGW, (WPARAM)0, (LPARAM)(LPWSTR)(szConfigName) ) #endif
```

```
#ifdef EE_STRICT inline BOOL Editor_DocSetConfigW( HWND hwnd, int iDoc, LPWSTR szConfigName ) { return (BOOL)SNDMSG
( (hwnd), EE_SET_CONFIGW, (WPARAM)MAKEWPARAM(0, (iDoc)+1), (LPARAM)(LPWSTR)(szConfigName) ); } #else #define
Editor_DocSetConfigW( hwnd, iDoc, szConfigName ) \ (BOOL)SNDMSG( (hwnd), EE_SET_CONFIGW, (WPARAM)MAKEWPARAM(0,
(iDoc)+1), (LPARAM)(LPWSTR)(szConfigName) ) #endif #define EE_EMPTY_UNDO_BUFFER (EE_FIRST+31) #ifdef EE_STRICT inline
void Editor_EmptyUndoBuffer( HWND hwnd ) { SNDMSG( (hwnd), EE_EMPTY_UNDO_BUFFER, (WPARAM)0, (LPARAM)0 ); } #else
#define Editor_EmptyUndoBuffer( hwnd ) \ (void)SNDMSG( (hwnd), EE_EMPTY_UNDO_BUFFER, (WPARAM)0, (LPARAM)0 ) #endif
#define OVERWRITE_PER_PROP 0 #define OVERWRITE_INSERT 1 #define OVERWRITE_OVERWRITE 2 #define
OVERWRITE_MASK 3 #define KEEP_SOURCE_RETURN_TYPE 0x00000000 #define KEEP_DEST_RETURN_TYPE 0x00000010 #define
EE_INSERT_STRINGA (EE_FIRST+32) // (int)wParam = nInsertType // (LPCSTR)lParam = szString inline void Editor_InsertStringA
( HWND hwnd, LPCSTR szString, bool bKeepDestReturnType = false ) { SNDMSG( (hwnd), EE_INSERT_STRINGA, (WPARAM)
OVERWRITE_PER_PROP | (bKeepDestReturnType ? KEEP_DEST_RETURN_TYPE : KEEP_SOURCE_RETURN_TYPE), (LPARAM)
(LPCSTR)(szString) ); } inline void Editor_InsertA( HWND hwnd, LPCSTR szString, bool bKeepDestReturnType = false ) { SNDMSG
( (hwnd), EE_INSERT_STRINGA, (WPARAM)OVERWRITE_INSERT | (bKeepDestReturnType ? KEEP_DEST_RETURN_TYPE :
KEEP_SOURCE_RETURN_TYPE), (LPARAM)(LPCSTR)(szString) ); } inline void Editor_OverwriteA( HWND hwnd, LPCSTR szString,
bool bKeepDestReturnType = false ) { SNDMSG( (hwnd), EE_INSERT_STRINGA, (WPARAM)OVERWRITE_OVERWRITE |
(bKeepDestReturnType ? KEEP_DEST_RETURN_TYPE : KEEP_SOURCE_RETURN_TYPE), (LPARAM)(LPCSTR)(szString) ); } #define
EE_INSERT_STRINGW (EE_FIRST+52) // (int)wParam = nInsertType | bKeepDestReturnType // (LPCWSTR)lParam = szString inline void
Editor_InsertStringW( HWND hwnd, LPCWSTR szString, bool bKeepDestReturnType = false ) { SNDMSG( (hwnd),
EE_INSERT_STRINGW, (WPARAM)OVERWRITE_PER_PROP | (bKeepDestReturnType ? KEEP_DEST_RETURN_TYPE :
KEEP_SOURCE_RETURN_TYPE), (LPARAM)(LPCWSTR)(szString) ); } inline void Editor_InsertW( HWND hwnd, LPCWSTR szString,
bool bKeepDestReturnType = false ) { SNDMSG( (hwnd), EE_INSERT_STRINGW, (WPARAM)OVERWRITE_INSERT |
(bKeepDestReturnType ? KEEP_DEST_RETURN_TYPE : KEEP_SOURCE_RETURN_TYPE), (LPARAM)(LPCWSTR)(szString) ); } inline
void Editor_OverwriteW( HWND hwnd, LPCWSTR szString, bool bKeepDestReturnType = false ) { SNDMSG( (hwnd),
EE_INSERT_STRINGW, (WPARAM)OVERWRITE_OVERWRITE | (bKeepDestReturnType ? KEEP_DEST_RETURN_TYPE :
KEEP_SOURCE_RETURN_TYPE), (LPARAM)(LPCWSTR)(szString) ); } #define EE_SET_SEL_VIEW (EE_FIRST+33) // (POINT_PTR*)
wParam = pptSelStart, (POINT_PTR*)lParam = pptSelEnd #ifdef EE_STRICT inline void Editor_SetSelView( HWND hwnd, POINT_PTR*
pptSelStart, POINT_PTR* pptSelEnd ) { SNDMSG( (hwnd), EE_SET_SEL_VIEW, (WPARAM)(POINT_PTR*)(pptSelStart), (LPARAM)
(POINT_PTR*)(pptSelEnd) ); } #else #define Editor_SetSelView( hwnd, pptSelStart, pptSelEnd ) \ (void)SNDMSG( (hwnd),
EE_SET_SEL_VIEW, (WPARAM)(POINT_PTR*)(pptSelStart), (LPARAM)(POINT_PTR*)(pptSelEnd) ) #endif #define EE_FINDA
(EE_FIRST+34) // (UINT)wParam = nFlags, (LPCSTR)lParam = szFind // returns (BOOL)bSuccess #ifdef EE_STRICT inline BOOL
Editor_FindA( HWND hwnd, UINT nFlags, LPCSTR szFind ) { return (BOOL)SNDMSG( (hwnd), EE_FINDA, (WPARAM)(UINT)(nFlags),
(LPARAM)(LPCSTR)(szFind) ); } #else #define Editor_FindA( hwnd, nFlags, szFind ) \ (BOOL)SNDMSG( (hwnd), EE_FINDA, (WPARAM)
(UINT)(nFlags), (LPARAM)(LPCSTR)(szFind) ) #endif #define EE_FINDW (EE_FIRST+53) // (UINT)wParam = nFlags, (LPCWSTR)lParam
= szFind // returns (BOOL)bSuccess #ifdef EE_STRICT inline BOOL Editor_FindW( HWND hwnd, UINT nFlags, LPCWSTR szFind )
{ return (BOOL)SNDMSG( (hwnd), EE_FINDW, (WPARAM)(UINT)(nFlags), (LPARAM)(LPCWSTR)(szFind) ); } #else #define
Editor_FindW( hwnd, nFlags, szFind ) \ (BOOL)SNDMSG( (hwnd), EE_FINDW, (WPARAM)(UINT)(nFlags), (LPARAM)(LPCWSTR)
(szFind) ) #endif #define EE_REPLACEA (EE_FIRST+35) // (UINT)wParam = nFlags, (LPCWSTR)lParam = szFindReplace // returns (BOOL)
bSuccess #ifdef EE_STRICT inline BOOL Editor_ReplaceA( HWND hwnd, UINT nFlags, LPCSTR szFindReplace ) { return (BOOL)SNDMSG
( (hwnd), EE_REPLACEA, (WPARAM)(UINT)(nFlags), (LPARAM)(LPCSTR)(szFindReplace) ); } #else #define Editor_ReplaceA( hwnd,
nFlags, szFindReplace ) \ (BOOL)SNDMSG( (hwnd), EE_REPLACEA, (WPARAM)(UINT)(nFlags), (LPARAM)(LPCSTR)(szFindReplace) )
#endif #define EE_REPLACEW (EE_FIRST+54) // (UINT)wParam = nFlags, (LPCWSTR)lParam = szFindReplace // returns (BOOL)bSuccess
#ifdef EE_STRICT inline BOOL Editor_ReplaceW( HWND hwnd, UINT nFlags, LPCWSTR szFindReplace ) { return (BOOL)SNDMSG
( (hwnd), EE_REPLACEW, (WPARAM)(UINT)(nFlags), (LPARAM)(LPCWSTR)(szFindReplace) ); } #else #define Editor_ReplaceW( hwnd,
nFlags, szFindReplace ) \ (BOOL)SNDMSG( (hwnd), EE_REPLACEW, (WPARAM)(UINT)(nFlags), (LPARAM)(LPCWSTR)
(szFindReplace) ) #endif #define EE_LOAD_CONFIGA (EE_FIRST+36) // (LPCSTR)lParam = szConfigName #ifdef EE_STRICT inline
BOOL Editor_LoadConfigA( HWND hwnd, LPCSTR szConfigName ) { return (BOOL)SNDMSG( (hwnd), EE_LOAD_CONFIGA,
(WPARAM)0, (LPARAM)(LPCSTR)(szConfigName) ); } #else #define Editor_LoadConfigA( hwnd, szConfigName ) \ (BOOL)SNDMSG
( (hwnd), EE_LOAD_CONFIGA, (WPARAM)0, (LPARAM)(LPCSTR)(szConfigName) ) #endif #define EE_LOAD_CONFIGW
(EE_FIRST+55) // (LPCWSTR)lParam = szConfigName #ifdef EE_STRICT inline BOOL Editor_LoadConfigW( HWND hwnd, LPCWSTR
szConfigName ) { return (BOOL)SNDMSG( (hwnd), EE_LOAD_CONFIGW, (WPARAM)0, (LPARAM)(LPCWSTR)(szConfigName) ); } #else
#define Editor_LoadConfigW( hwnd, szConfigName ) \ (BOOL)SNDMSG( (hwnd), EE_LOAD_CONFIGW, (WPARAM)0, (LPARAM)
(LPCWSTR)(szConfigName) ) #endif #define EE_SET_STATUSA (EE_FIRST+37) // (LPCSTR)lParam = szStatus #ifdef EE_STRICT inline
void Editor_SetStatusA( HWND hwnd, LPCSTR szStatus ) { SNDMSG( (hwnd), EE_SET_STATUSA, (WPARAM)0, (LPARAM)(LPCSTR)
(szStatus) ); } #else #define Editor_SetStatusA( hwnd, szStatus ) \ (void)SNDMSG( (hwnd), EE_SET_STATUSA, (WPARAM)0, (LPARAM)
(LPCSTR)(szStatus) ) #endif #define EE_SET_STATUSW (EE_FIRST+56) // (LPCWSTR)lParam = szStatus #ifdef EE_STRICT inline void
Editor_SetStatusW( HWND hwnd, LPCWSTR szStatus ) { SNDMSG( (hwnd), EE_SET_STATUSW, (WPARAM)0, (LPARAM)(LPCWSTR)
(szStatus) ); } #else #define Editor_SetStatusW( hwnd, szStatus ) \ (void)SNDMSG( (hwnd), EE_SET_STATUSW, (WPARAM)0, (LPARAM)
(LPCWSTR)(szStatus) ) #endif #define EE_CONVERT (EE_FIRST+38) // (UINT)wParam = nFlags #ifdef EE_STRICT inline BOOL
Editor_Convert( HWND hwnd, UINT nFlags ) { return (BOOL)SNDMSG( (hwnd), EE_CONVERT, (WPARAM)(UINT)(nFlags), (LPARAM)
0 ); } #else #define Editor_Convert( hwnd, nFlags ) \ (BOOL)SNDMSG( (hwnd), EE_CONVERT, (WPARAM)(UINT)(nFlags), (LPARAM)0 )
#endif #define EE_GET_MARGIN (EE_FIRST+39) // returns (UINT_PTR)nMaxMargin #ifdef EE_STRICT inline UINT_PTR
Editor_GetMargin( HWND hwnd ) { return (UINT_PTR)SNDMSG( (hwnd), EE_GET_MARGIN, (WPARAM)0, (LPARAM)0 ); } #else #define
Editor_GetMargin( hwnd ) \ (UINT_PTR)SNDMSG( (hwnd), EE_GET_MARGIN, (WPARAM)0, (LPARAM)0 ) #endif #define
EE_GET_VERSION (EE_FIRST+40) // (UINT*)wParam = pnProductType // returns (UINT)nVersion #ifdef EE_STRICT inline UINT
Editor_GetVersion( HWND hwnd ) { return (UINT)SNDMSG( (hwnd), EE_GET_VERSION, (WPARAM)0, (LPARAM)0 ); } #else #define
Editor_GetVersion( hwnd ) \ (UINT)SNDMSG( (hwnd), EE_GET_VERSION, (WPARAM)0, (LPARAM)0 ) #endif #ifdef EE_STRICT inline
UINT Editor_GetVersionEx( HWND hwnd, UINT* pnProductType ) { return (UINT)SNDMSG( (hwnd), EE_GET_VERSION, (WPARAM)
pnProductType, (LPARAM)0 ); } #else #define Editor_GetVersionEx( hwnd, pnProductType ) \ (UINT)SNDMSG( (hwnd),
EE_GET_VERSION, (WPARAM)(UINT*)pnProductType, (LPARAM)0 ) #endif #define VERSION_FREE 4 #define VERSION_LITE 3
#define VERSION_PRO 2 #define VERSION_STD 1 #define EE_GET_REF (EE_FIRST+41) // (ATOM)lParam = atom // return (int)nRef
#ifdef EE_STRICT inline int Editor_GetRef( HWND hwnd, ATOM atom ) { return (int)SNDMSG( (hwnd), EE_GET_REF, (WPARAM)0,
(LPARAM)(ATOM)atom ); } #else #define Editor_GetRef( hwnd, atom ) \ (int)SNDMSG( (hwnd), EE_GET_REF, (WPARAM)0, (LPARAM)
(ATOM)atom ) #endif #define EE_ADD_REF (EE_FIRST+42) // (HINSTANCE)lParam = hInstance // return (int)nRef #ifdef EE_STRICT
inline int Editor_AddRef( HWND hwnd, HINSTANCE hInstance ) { return (int)SNDMSG( (hwnd), EE_ADD_REF, (WPARAM)0, (LPARAM)
(HINSTANCE)hInstance ); } #else #define Editor_AddRef( hwnd, hInstance ) \ (int)SNDMSG( (hwnd), EE_ADD_REF, (WPARAM)0,
(LPARAM)(HINSTANCE)hInstance ) #endif #define EE_RELEASE (EE_FIRST+43) // (HINSTANCE)lParam = hInstance // return (int)nRef
#ifdef EE_STRICT inline int Editor_Release( HWND hwnd, HINSTANCE hInstance ) { return (int)SNDMSG( (hwnd), EE_RELEASE,
(WPARAM)0, (LPARAM)(HINSTANCE)hInstance ); } #else #define Editor_Release( hwnd, hInstance ) \ (int)SNDMSG( (hwnd),
EE_RELEASE, (WPARAM)0, (LPARAM)(HINSTANCE)hInstance ) #endif #define EE_REDRAW (EE_FIRST+44) // (BOOL)wParam =
bRedraw #ifdef EE_STRICT inline void Editor_Redraw( HWND hwnd, BOOL bRedraw ) { SNDMSG( (hwnd), EE_REDRAW, (WPARAM)
bRedraw, (LPARAM)0 ); } #else #define Editor_Redraw( hwnd, bRedraw ) \ (void)SNDMSG( (hwnd), EE_REDRAW, (WPARAM)(BOOL)
bRedraw, (LPARAM)0 ) #endif #define EE_GET_SEL_TYPE (EE_FIRST+45) // (BOOL)wParam = bNeedAlways // return (int)nSelType
#ifdef EE_STRICT inline int Editor_GetSelType( HWND hwnd ) { return (int)SNDMSG( (hwnd), EE_GET_SEL_TYPE, (WPARAM)0,
```

(LPARAM)0); } inline int Editor_GetSelTypeEx(HWND hwnd, BOOL bNeedAlways) { return (int)SNDMSG((hwnd), EE_GET_SEL_TYPE, (WPARAM)(BOOL)bNeedAlways, (LPARAM)0); } #else #define Editor_GetSelType(hwnd) \ (int)SNDMSG((hwnd), EE_GET_SEL_TYPE, (WPARAM)0, (LPARAM)0) #define Editor_GetSelTypeEx(hwnd, bNeedAlways) \ (int)SNDMSG((hwnd), EE_GET_SEL_TYPE, (WPARAM)(BOOL)bNeedAlways, (LPARAM)0) #endif #define EE_IS_CHAR_HALF_OR_FULL (EE_FIRST+57) // (WCHAR)wParam = ch // return (int)nWidth #ifdef EE_STRICT inline int Editor_IsCharHalfOrFull(HWND hwnd, WCHAR ch) { return (int)SNDMSG((hwnd), EE_IS_CHAR_HALF_OR_FULL, (WPARAM)ch, (LPARAM)0); } #else #define Editor_IsCharHalfOrFull(hwnd, ch) \ (int)SNDMSG ((hwnd), EE_IS_CHAR_HALF_OR_FULL, (WPARAM)ch, (LPARAM)0) #endif #define EE_INFO (EE_FIRST+58) // (int)wParam = nCmd // lParam = lParam // return lResult #ifdef EE_STRICT inline LRESULT Editor_Info(HWND hwnd, WPARAM nCmd, LPARAM lParam) { return (LRESULT)SNDMSG((hwnd), EE_INFO, (WPARAM)nCmd, (LPARAM)lParam); } #else #define Editor_Info(hwnd, nCmd, lParam) \ (LRESULT)SNDMSG((hwnd), EE_INFO, (WPARAM)nCmd, (LPARAM)lParam) #endif #ifdef EE_STRICT inline LRESULT Editor_DocInfo(HWND hwnd, int iDoc, WPARAM nCmd, LPARAM lParam) { return (LRESULT)SNDMSG((hwnd), EE_INFO, (WPARAM)MAKEWPARAM((nCmd),(iDoc+1)), (LPARAM)(lParam)); } #else #define Editor_DocInfo(hwnd, iDoc, nCmd, lParam) \ (LRESULT)SNDMSG((hwnd), EE_INFO, (WPARAM)MAKEWPARAM((nCmd),(iDoc+1)), (LPARAM)(lParam)) #endif #define EE_FREE (EE_FIRST+59) // (ATOM)lParam = atom // return (BOOL)bSuccess #ifdef EE_STRICT inline BOOL Editor_Free(HWND hwnd, ATOM atom) { return (BOOL)SNDMSG((hwnd), EE_FREE, (WPARAM)0, (LPARAM)(ATOM)atom); } #else #define Editor_Free(hwnd, atom) \ (BOOL)SNDMSG((hwnd), EE_FREE, (WPARAM)0, (LPARAM)(ATOM)atom) #endif #define EE_SET_SEL_TYPE (EE_FIRST+60) // (BOOL)wParam = bNeedAlways // (UINT)lParam = nSelType #ifdef EE_STRICT inline void Editor_SetSelType(HWND hwnd, UINT nSelType) { SNDMSG((hwnd), EE_SET_SEL_TYPE, (WPARAM)0, (LPARAM)nSelType); } inline void Editor_SetSelTypeEx(HWND hwnd, BOOL bNeedAlways, UINT nSelType) { SNDMSG((hwnd), EE_SET_SEL_TYPE, (WPARAM)(BOOL)bNeedAlways, (LPARAM) nSelType); } #else #define Editor_SetSelType(hwnd, nSelType) \ (void)SNDMSG((hwnd), EE_SET_SEL_TYPE, (WPARAM)0, (LPARAM) nSelType) #define Editor_SetSelTypeEx(hwnd, nSelType) \ (void)SNDMSG((hwnd), EE_SET_SEL_TYPE_EX, (WPARAM)(BOOL) bNeedAlways, (LPARAM)nSelType) #endif #define EE_GET_STATUSA (EE_FIRST+61) // (UINT_PTR)wParam = nBufSize // (LPSTR) lParam = szStatus // return (UINT_PTR)cchRequiredSize #ifdef EE_STRICT inline UINT_PTR Editor_GetStatusA(HWND hwnd, LPCSTR szStatus, UINT_PTR nBufSize) { return (UINT_PTR)SNDMSG((hwnd), EE_GET_STATUSA, (WPARAM)nBufSize, (LPARAM)(LPCSTR) (szStatus)); } #else #define Editor_GetStatusA(hwnd, szStatus, nBufSize) \ (UINT_PTR)SNDMSG((hwnd), EE_GET_STATUSA, (WPARAM)nBufSize, (LPARAM)(LPCSTR)(szStatus)) #endif #define EE_GET_STATUSW (EE_FIRST+62) // (UINT_PTR)wParam = nBufSize // (LPWSTR)lParam = szStatus // return (UINT_PTR)cchRequiredSize #ifdef EE_STRICT inline UINT_PTR Editor_GetStatusW (HWND hwnd, LPCWSTR szStatus, UINT_PTR nBufSize) { return (UINT_PTR)SNDMSG((hwnd), EE_GET_STATUSW, (WPARAM) nBufSize, (LPARAM)(LPCWSTR)(szStatus)); } #else #define Editor_GetStatusW(hwnd, szStatus, nBufSize) \ (UINT_PTR)SNDMSG ((hwnd), EE_GET_STATUSW, (WPARAM)nBufSize, (LPARAM)(LPCWSTR)(szStatus)) #endif #define EE_INSERT_FILEA (EE_FIRST+63) // (LOAD_FILE_INFO_EX*)wParam = plfi // (LPCSTR)lParam = szFileName // returns non-zero if success, 0 if failed #ifdef EE_STRICT inline BOOL Editor_InsertFileA(HWND hwnd, LOAD_FILE_INFO_EX* plfi, LPCSTR szFileName) { return (BOOL)SNDMSG ((hwnd), EE_INSERT_FILEA, (WPARAM)plfi, (LPARAM)(LPCSTR)(szFileName)); } #else #define Editor_InsertFileA(hwnd, plfi, szFileName) \ (BOOL)SNDMSG((hwnd), EE_INSERT_FILEA, (WPARAM)(LOAD_FILE_INFO_EX*)plfi, (LPARAM)(LPCSTR) (szFileName)) #endif #define EE_INSERT_FILEW (EE_FIRST+64) // (LOAD_FILE_INFO_EX*)wParam = plfi // (LPCWSTR)lParam = szFileName // returns non-zero if success, 0 if failed #ifdef EE_STRICT inline BOOL Editor_InsertFileW(HWND hwnd, LOAD_FILE_INFO_EX* plfi, LPCWSTR szFileName) { return (BOOL)SNDMSG((hwnd), EE_INSERT_FILEW, (WPARAM) (LOAD_FILE_INFO_EX*)plfi, (LPARAM)(LPCWSTR)(szFileName)); } #else #define Editor_InsertFileW(hwnd, plfi, szFileName) \ (BOOL) SNDMSG((hwnd), EE_INSERT_FILEW, (WPARAM)plfi, (LPARAM)(LPCWSTR)(szFileName)) #endif #define EE_FIND_IN_FILESA (EE_FIRST+65) // wParam = 0 // (GREP_INFOA*)lParam = pGrepInfo #ifdef EE_STRICT inline BOOL Editor_FindInFilesA(HWND hwnd, GREP_INFOA* pGrepInfo) { return (BOOL)SNDMSG((hwnd), EE_FIND_IN_FILESA, (WPARAM)0, (LPARAM)(GREP_INFOA*) pGrepInfo); } #else #define Editor_FindInFilesA(hwnd, pGrepInfo) \ (BOOL)SNDMSG((hwnd), EE_FIND_IN_FILESA, (WPARAM)0, (LPARAM)pGrepInfo) #endif #define EE_FIND_IN_FILESW (EE_FIRST+66) // wParam = 0 // (GREP_INFOW*)lParam = pGrepInfo #ifdef EE_STRICT inline BOOL Editor_FindInFilesW(HWND hwnd, GREP_INFOW* pGrepInfo) { return (BOOL)SNDMSG((hwnd), EE_FIND_IN_FILESW, (WPARAM)0, (LPARAM)pGrepInfo); } #else #define Editor_FindInFilesW(hwnd, pGrepInfo) \ (BOOL)SNDMSG ((hwnd), EE_FIND_IN_FILESW, (WPARAM)0, (LPARAM)(GREP_INFOW*)pGrepInfo) #endif #define EE_REPLACE_IN_FILESA (EE_FIRST+67) // wParam = 0 // (GREP_INFOA*)lParam = pGrepInfo #ifdef EE_STRICT inline BOOL Editor_ReplaceInFilesA(HWND hwnd, GREP_INFOA* pGrepInfo) { return (BOOL)SNDMSG((hwnd), EE_REPLACE_IN_FILESA, (WPARAM)0, (LPARAM) pGrepInfo); } #else #define Editor_ReplaceInFilesA(hwnd, pGrepInfo) \ (BOOL)SNDMSG((hwnd), EE_REPLACE_IN_FILESA, (WPARAM)0, (LPARAM)(GREP_INFOA*)pGrepInfo) #endif #define EE_REPLACE_IN_FILESW (EE_FIRST+68) // wParam = 0 // (GREP_INFOW*)lParam = pGrepInfo #ifdef EE_STRICT inline BOOL Editor_ReplaceInFilesW(HWND hwnd, GREP_INFOW* pGrepInfo) { return (BOOL)SNDMSG(hwnd, EE_REPLACE_IN_FILESW, (WPARAM)0, (LPARAM)pGrepInfo); } #else #define Editor_ReplaceInFilesW(hwnd, pGrepInfo) \ (BOOL)SNDMSG((hwnd), EE_REPLACE_IN_FILESW, (WPARAM)0, (LPARAM) (GREP_INFOW*)pGrepInfo) #endif #define EE_GET_ANCHOR_POS (EE_FIRST+69) // (int)wParam = nLogical // (POINT_PTR*)lParam = pptPos #ifdef EE_STRICT inline void Editor_GetAnchorPos(HWND hwnd, int nLogical, POINT_PTR* pptPos) { SNDMSG((hwnd), EE_GET_ANCHOR_POS, (WPARAM)(int)(nLogical), (LPARAM)(POINT_PTR*)(pptPos)); } #else #define Editor_GetAnchorPos(hwnd, nLogical, pptPos) \ (void)SNDMSG((hwnd), EE_GET_ANCHOR_POS, (WPARAM)(int)(nLogical), (LPARAM)(POINT_PTR*)(pptPos)) #endif #define EE_SET_ANCHOR_POS (EE_FIRST+70) // (int)wParam = nLogical // (POINT_PTR*)lParam = pptPos #ifdef EE_STRICT inline void Editor_SetAnchorPos(HWND hwnd, int nLogical, POINT_PTR* pptPos) { SNDMSG((hwnd), EE_SET_ANCHOR_POS, (WPARAM)(int)(nLogical), (LPARAM)(POINT_PTR*)(pptPos)); } #else #define Editor_SetAnchorPos(hwnd, nLogical, pptPos) \ (void) SNDMSG((hwnd), EE_SET_ANCHOR_POS, (WPARAM)(int)(nLogical), (LPARAM)(POINT_PTR*)(pptPos)) #endif #define EE_GET_REDRAW (EE_FIRST+71) #ifdef EE_STRICT inline BOOL Editor_GetRedraw(HWND hwnd) { return (BOOL)SNDMSG ((hwnd), EE_GET_REDRAW, (WPARAM)0, (LPARAM)0); } #else #define Editor_GetRedraw(hwnd) \ (BOOL)SNDMSG((hwnd), EE_GET_REDRAW, (WPARAM)0, (LPARAM)0) #endif #define EE_DO_IDLE (EE_FIRST+72) #ifdef EE_STRICT inline void Editor_DoIdle(HWND hwnd, BOOL bResetTab) { SNDMSG((hwnd), EE_DO_IDLE, (WPARAM)(bResetTab), (LPARAM)0); } #else #define Editor_DoIdle(hwnd, bResetTab) \ (void)SNDMSG((hwnd), EE_DO_IDLE, (WPARAM)(bResetTab), (LPARAM)0) #endif #define EE_CUSTOM_BAR_OPEN (EE_FIRST+73) // (CUSTOM_BAR_INFO*)lParam = pCustomBarInfo #ifdef EE_STRICT inline UINT Editor_CustomBarOpen(HWND hwnd, CUSTOM_BAR_INFO* pCustomBarInfo) { return (UINT)SNDMSG((hwnd), EE_CUSTOM_BAR_OPEN, 0, (LPARAM)pCustomBarInfo); } #else #endif #define EE_CUSTOM_BAR_CLOSE (EE_FIRST+74) // (int) wParam = nCustomBarID #ifdef EE_STRICT inline BOOL Editor_CustomBarClose(HWND hwnd, UINT nCustomBarID) { return (BOOL) SNDMSG((hwnd), EE_CUSTOM_BAR_CLOSE, (WPARAM)nCustomBarID, 0); } #else #endif #define EE_MATCH_REGEX (EE_FIRST+75) #ifdef EE_STRICT inline BOOL Editor_MatchRegex(HWND hwnd, MATCH_REGEX_INFO* pMatchRegexInfo) { return (BOOL)SNDMSG((hwnd), EE_MATCH_REGEX, (WPARAM)0, (LPARAM)pMatchRegexInfo); } #else #define Editor_MatchRegex(hwnd, pMatchRegexInfo) \ (BOOL)SNDMSG((hwnd), EE_MATCH_REGEX, (WPARAM)0, (LPARAM)pMatchRegexInfo) #endif #define EE_FIND_REGEX (EE_FIRST+76) #ifdef EE_STRICT inline BOOL Editor_FindRegex(HWND hwnd, FIND_REGEX_INFO* pFindRegexInfo) { return (BOOL)SNDMSG((hwnd), EE_FIND_REGEX, (WPARAM)0, (LPARAM)pFindRegexInfo); } #else #define Editor_FindRegex(hwnd, pFindRegexInfo) \ (BOOL)SNDMSG((hwnd), EE_FIND_REGEX, (WPARAM)0, (LPARAM)pFindRegexInfo) #endif #define EE_GET_OUTLINE_LEVEL (EE_FIRST+77) // (INT_PTR)wParam = nLogicalLine // return nLevel inline int Editor_GetOutlineLevel(HWND hwnd, INT_PTR nLogicalLine) { return (int)SNDMSG((hwnd), EE_GET_OUTLINE_LEVEL, (WPARAM) nLogicalLine, (LPARAM)0); } #define EE_SET_OUTLINE_LEVEL (EE_FIRST+78) // (INT_PTR)wParam = nLogicalLine // (int)lParam = nLevel inline void Editor_SetOutlineLevel(HWND hwnd, INT_PTR nLogicalLine, int nLevel) { SNDMSG((hwnd), EE_SET_OUTLINE_LEVEL, (WPARAM)nLogicalLine, (LPARAM)nLevel); } #define EE_SHOW_OUTLINE (EE_FIRST+79) // (UINT) wParam = nFlags // return lResult inline void Editor_ShowOutline(HWND hwnd, WPARAM nFlags) { SNDMSG((hwnd),

EE_SHOW_OUTLINE, (WPARAM)nFlags, (LPARAM)0); } #define SHOW_OUTLINE_SHOW 1 #define SHOW_OUTLINE_HIDE 0 #define EE_ENUM_CONFIG (EE_FIRST+80) // (size_t)wParam = cchBuf // (LPWSTR)lParam = pBuf // return nSize inline size_t Editor_EnumConfig(HWND hwnd, LPWSTR pBuf, size_t cchBuf) { return (size_t)SNDMSG((hwnd), EE_ENUM_CONFIG, (WPARAM) cchBuf, (LPARAM)pBuf); } #define EE_TOOLBAR_OPEN (EE_FIRST+81) // (TOOLBAR_INFO*)lParam = pToolbarInfo inline UINT Editor_ToolbarOpen(HWND hwnd, TOOLBAR_INFO* pToolbarInfo) { return (UINT)SNDMSG((hwnd), EE_TOOLBAR_OPEN, 0, (LPARAM)pToolbarInfo); } #define EE_TOOLBAR_CLOSE (EE_FIRST+82) // (UINT)wParam = nToolbarID inline BOOL Editor_ToolbarClose(HWND hwnd, UINT nCustomRebarID) { return (BOOL)SNDMSG((hwnd), EE_TOOLBAR_CLOSE, (WPARAM) nCustomRebarID, 0); } #define EE_TOOLBAR_SHOW (EE_FIRST+83) // (UINT)wParam = nToolbarID // (BOOL)lParam = bVisible inline BOOL Editor_ToolbarShow(HWND hwnd, UINT nCustomRebarID, BOOL bVisible) { return (BOOL)SNDMSG((hwnd), EE_TOOLBAR_SHOW, (WPARAM)nCustomRebarID, (LPARAM)bVisible); } #define EE_HELP (EE_FIRST+84) // (UINT)wParam = nFlag (must be 0) // (LPCTSTR)lParam = szPageURL inline void Editor_Help(HWND hwnd, LPCTSTR szPageURL) { SNDMSG((hwnd), EE_HELP, 0, (LPARAM)szPageURL); } #define EE_REG_SET_VALUE (EE_FIRST+85) // (REG_SET_VALUE_INFO*)lParam = pRegSetValueInfo; inline LONG Editor_RegSetValue(HWND hwnd, DWORD dwKey, LPCWSTR pszConfig, LPCWSTR pszValue, DWORD dwType, const BYTE* lpData, DWORD cbData, DWORD dwFlags) { REG_SET_VALUE_INFO info = { 0 }; info.cbSize = sizeof(info); info.dwKey = dwKey; info.pszConfig = pszConfig; info.pszValue = pszValue; info.dwType = dwType; info.lpData = lpData; info.cbData = cbData; info.dwFlags = dwFlags; return (LONG)SNDMSG((hwnd), EE_REG_SET_VALUE, 0, (LPARAM)&info); } #define EE_REG_QUERY_VALUE (EE_FIRST+86) // (REG_QUERY_VALUE_INFO*)lParam = pRegQueryValueInfo; inline LONG Editor_RegQueryValue(HWND hwnd, DWORD dwKey, LPCWSTR pszConfig, LPCWSTR pszValue, DWORD dwType, BYTE* lpData, DWORD* lpcbData, DWORD dwFlags) { REG_QUERY_VALUE_INFO info = { 0 }; info.cbSize = sizeof(info); info.dwKey = dwKey; info.pszConfig = pszConfig; info.pszValue = pszValue; info.dwType = dwType; info.lpData = lpData; info.lpcbData = lpcbData; info.dwFlags = dwFlags; return (LONG)SNDMSG((hwnd), EE_REG_QUERY_VALUE, 0, (LPARAM)&info); } #define EE_QUERY_STRING (EE_FIRST+87) // (UINT)wParam = nCmdID, (LPWSTR)lParam = psz (buffer must be MAX_PATH) // returns (BOOL)bValidCmd inline BOOL Editor_QueryString(HWND hwnd, UINT nCmdID, LPWSTR psz) { return (BOOL)SNDMSG(hwnd, EE_QUERY_STRING, (WPARAM)nCmdID, (LPARAM)psz); } #define EE_KEYBOARD_PROP (EE_FIRST+88) // (UINT)wParam = nCmdID, (LPCWSTR)lParam = pszConfigName // returns (BOOL)bResult inline BOOL Editor_KeyboardProp(HWND hwnd, UINT nCmdID, LPCWSTR pszConfigName) { return (BOOL)SNDMSG(hwnd, EE_KEYBOARD_PROP, (WPARAM)nCmdID, (LPARAM)pszConfigName); } #define EE_GET_ACCEL_ARRAY (EE_FIRST+89) // (UINT)wParam = nBufSize (size of buffer in ACCEL) // (ACCEL*)lParam = pAccel inline UINT Editor_GetAccelArray(HWND hwnd, ACCEL* pAccel, UINT nBufSize) { return (UINT)SNDMSG(hwnd, EE_GET_ACCEL_ARRAY, (WPARAM)nBufSize, (LPARAM)pAccel); } #define EE_OUTPUT_STRING (EE_FIRST+90) // (UINT)wParam = flags // (LPCWSTR)lParam = pszString inline BOOL Editor_OutputString(HWND hwnd, LPCWSTR pszString, UINT flags) { return (BOOL)SNDMSG(hwnd, EE_OUTPUT_STRING, (WPARAM)flags, (LPARAM)pszString); } #define EE_OUTPUT_DIR (EE_FIRST+91) // (LPCWSTR)lParam = pszCurrDir inline BOOL Editor_OutputDir(HWND hwnd, LPCWSTR pszCurrDir) { return (BOOL)SNDMSG(hwnd, EE_OUTPUT_DIR, (WPARAM)0, (LPARAM)pszCurrDir); } #define EE_ENUM_HIGHLIGHT (EE_FIRST+92) // (LPWSTR)pBuf // (size_t)cchBuf inline size_t Editor_EnumHighlight(HWND hwnd, LPWSTR pBuf, size_t cchBuf) { return (size_t)SNDMSG((hwnd), EE_ENUM_HIGHLIGHT, (WPARAM)cchBuf, (LPARAM)pBuf); } // #define EE_LAST (EE_FIRST+255) #define HIGHLIGHT_COLOR_MASK 0x000f #define HIGHLIGHT_WORD 0x0100 #define HIGHLIGHT_RIGHT_SIDE 0x0200 #define HIGHLIGHT_INSIDE_TAG 0x0400 #define HIGHLIGHT_REG_EXP 0x0800 #define HIGHLIGHT_CASE 0x1000 #define HIGHLIGHT_RIGHT_ALL 0x2000 #define FLAG_OPEN_OUTPUT 1 #define FLAG_CLOSE_OUTPUT 2 #define FLAG_FOCUS_OUTPUT 4 #define FLAG_CLEAR_OUTPUT 8 #define EI_GET_ENCODE 256 #define EI_SET_ENCODE 257 #define EI_GET_SIGNATURE 268 #define EI_SET_SIGNATURE 269 #define EI_GET_FONT_CHARSET 270 #define EI_SET_FONT_CHARSET 271 #define EI_GET_FONT_CP 272 #define EI_GET_INPUT_CP 274 #define EI_GET_SHOW_TAG 276 #define EI_SET_SHOW_TAG 277 #define EI_GET_FILE_NAMEA 278 #define EI_GET_FILE_NAMEW 280 #define EI_IS_PROPORTIONAL_FONT 282 #define EI_GET_NEXT_BOOKMARK 284 #define EI_GET_HILITE_FIND 286 #define EI_SET_HILITE_FIND 287 #define EI_GET_APP_VERSIONA 288 #define EI_GET_APP_VERSIONW 290 #define EI_GET_READ_ONLY 296 #define EI_IS_WINDOW_COMBINED 298 #define EI_WINDOW_COMBINE 299 #define EI_IS_UNDO_COMBINED 300 // new from v5.00 #define EI_GET_DOC_COUNT 301 #define EI_INDEX_TO_DOC 302 #define EI_DOC_TO_INDEX 303 #define EI_ZORDER_TO_DOC 304 #define EI_DOC_TO_ZORDER 305 #define EI_GET_ACTIVE_INDEX 306 #define EI_SET_ACTIVE_INDEX 307 #define EI_GET_FULL_TITLEA 308 #define EI_GET_FULL_TITLEW 309 #define EI_GET_SHORT_TITLEA 310 #define EI_GET_SHORT_TITLEW 311 #define EI_GET_SAVE_AS_TITLEA 312 #define EI_GET_SAVE_AS_TITLEW 313 #define EI_MOVE_ORDER 314 #define EI_CLOSE_DOC 315 #define EI_SAVE_DOC 316 // new from v7.00 #define EI_GET_CURRENT_FOLDER 317 #define EI_IS_LARGE_DOC 318 #define POS_VIEW 0 #define POS_LOGICAL_A 1 #define POS_LOGICAL_W 2 #define POS_TAB_A 3 #define MAX_LINE_COLUMN_MODE 4 #define POS_SCROLL_DONT_CARE 0x00000000 #define POS_SCROLL_CENTER 0x00000010 #define POS_SCROLL_TOP 0x00000020 #define POS_WANT_X 0x00010000 #define POS_WANT_Y 0x00020000 #define SEL_TYPE_NONE 0 #define SEL_TYPE_CHAR 1 #define SEL_TYPE_LINE 2 #define SEL_TYPE_BOX 3 #define SEL_TYPE_MASK 0x000f #define SEL_TYPE_KEYBOARD 0x0010 #define SEL_TYPE_SELECTED 0x0020 // EE_SAVE_FILE #define DO_SAVE_NONE 0 #define DO_SAVE_REPLACE 1 #define DO_SAVE_RENAME 2 // EE_CONVERT #define FLAG_MAKE_LOWER 0 #define FLAG_MAKE_UPPER 1 #define FLAG_HAN_TO_ZEN 2 #define FLAG_ZEN_TO_HAN 3 #define FLAG_CAPITALIZE 4 #define FLAG_CONVERT_MASK 7 #define FLAG_CONVERT_SELECT_ALL 0x0100 #define FLAG_CONVERT_KATA 0x0400 #define FLAG_CONVERT_ALPHANUMERIC 0x0800 #define FLAG_CONVERT_MARK 0x1000 #define FLAG_CONVERT_SPACE 0x2000 #define FLAG_CONVERT_KANA_MARK 0x4000 #define FLAG_CONVERT_ALL_TYPES 0xfe00 // EE_FIND, EE_REPLACE, EE_FIND_IN_FILES, EE_REPLACE_IN_FILES, EE_MATCH_REGEX, EE_FIND_REGEX #define FLAG_FIND_NEXT 0x0001 // EE_FIND only #define FLAG_FIND_CASE 0x0002 // EE_FIND, EE_REPLACE, EE_MATCH_REGEX and EE_FIND_REGEX #define FLAG_FIND_ESCAPE 0x0004 // EE_FIND and EE_REPLACE #define FLAG_REPLACE_SEL_ONLY 0x0008 // EE_REPLACE only #define FLAG_REPLACE_ALL 0x0010 // EE_REPLACE only #define FLAG_FIND_NO_PROMPT 0x0020 // EE_FIND and EE_REPLACE #define FLAG_FIND_ONLY_WORD 0x0040 // EE_FIND, EE_REPLACE and EE_FIND_REGEX #define FLAG_FIND_AROUND 0x0080 // EE_FIND only #define FLAG_FIND_REG_EXP 0x0100 // EE_FIND and EE_REPLACE #define FLAG_FIND_CLOSE 0x0200 // EE_FIND and EE_REPLACE #define FLAG_FIND_RECURSIVE 0x0400 // EE_FIND_IN_FILES and EE_REPLACE_IN_FILES #define FLAG_FIND_FILENAMES_ONLY 0x0800 // EE_FIND_IN_FILES only #define FLAG_REPLACE_KEEP_OPEN 0x1000 // EE_REPLACE_IN_FILES only #define FLAG_REPLACE_BACKUP 0x2000 // EE_REPLACE_IN_FILES only #define FLAG_FIND_IGNORE_FILES 0x4000 // EE_FIND_IN_FILES and EE_REPLACE_IN_FILES #define FLAG_FIND_OPEN_DOC 0x8000 // EE_FIND only v6.00 #define FLAG_GREP_MASK 0x7c00 #define FLAG_FIND_MASK 0xffff // GET_LINE_INFO #define FLAG_LOGICAL 1 #define FLAG_WITH_CRLF 2 #define UNINSTALL_FALSE 0 #define UNINSTALL_SIMPLE_DELETE 1 #define UNINSTALL_RUN_COMMAND 2 #define EP_FIRST (WM_USER+0x500) #define EP_QUERY_PROPERTIES (EP_FIRST+0) #define EP_SET_PROPERTIES (EP_FIRST+1) #define EP_GET_NAMEA (EP_FIRST+2) #define EP_GET_NAMEW (EP_FIRST+3) #define EP_GET_VERSIONA (EP_FIRST+4) #define EP_GET_VERSIONW (EP_FIRST+5) #define EP_QUERY_UNINSTALL (EP_FIRST+6) #define EP_SET_UNINSTALL (EP_FIRST+7) #define EP_GET_BITMAP (EP_FIRST+8) #define EP_GET_MASK (EP_FIRST+9) #define EP_GET_INFO (EP_FIRST+10) #define EP_PRE_TRANSLATE_MSG (EP_FIRST+11) #define EP_LAST (EP_FIRST+50) #ifdef _UNICODE #define EP_GET_NAME EP_GET_NAMEW #define EP_GET_VERSION EP_GET_VERSIONW #else #define EP_GET_NAME EP_GET_NAMEA #define EP_GET_VERSION EP_GET_VERSIONA #endif // EP_GET_BITMAP #define BITMAP_SMALL 0x00000000 #define BITMAP_LARGE 0x00000001 #define BITMAP_SIZE_MASK 0x0000000f #define BITMAP_16_COLOR 0x00000000 #define BITMAP_256_COLOR 0x00000010 #define BITMAP_24BIT_COLOR 0x00000020 #define BITMAP_COLOR_MASK 0x000000f0 #define BITMAP_DEFAULT 0x00000000 #define BITMAP_DISABLED 0x00000100 #define BITMAP_HOT 0x00000200 #define BITMAP_STATUS_MASK 0x00000f00 // EP_GET_INFO #define EPGI_ALLOW_OPEN_SAME_GROUP 1 #define EPGI_MAX_EE_VERSION 2 #define EPGI_MIN_EE_VERSION 3 #define EPGI_SUPPORT_EE_PRO 4 #define EPGI_SUPPORT_EE_STD

5 #define EPGI_ALLOW_MULTIPLE_INSTANCES 6 #define CHECK_FILE_CHANGED_NONE 0 #define CHECK_FILE_CHANGED_PROMPT 1 #define CHECK_FILE_CHANGED_AUTO 2 #define CHECK_FILE_CHANGED_EXCLUSIVE 3 #define MAX_CHECK_FILE_CHANGED 4 #define MIN_MONITOR_INTERVAL 1 #define MAX_MONITOR_INTERVAL 255 // inclusive #define DEF_MONITOR_INTERVAL 5 #define SIZE_OF_CUSTOMIZE_INFO 6660 #define MIN_SMOOTH_SCROLL_SPEED 1 #define MAX_SMOOTH_SCROLL_SPEED 9 #define DEF_SMOOTH_SCROLL_SPEED 5 class CCustomizeInfo { public: LOGFONTW m_alfScreen [MAX_CHARSET]; // screen fonts LOGFONTW m_alfPrint[MAX_CHARSET]; // printer fonts POINT m_ptShowScrollBar; // scroll bars (x: horizontal and y: vertical) 0: no display, 1: display only when necessary, 2: display always int m_nPrinterMarginTop; // printer top margin int m_nPrinterMarginBottom; // printer bottom margin int m_nPrinterMarginLeft; // printer left margin int m_nWrapMode; // wrap mode int m_nMarginNormal; // normal line margin int m_nMarginQuote; // quoted line margin int m_nTabSpace; // tab columns int m_nEncodingRead; // encoding for read BYTE m_nLineSpace; // line space BYTE m_nCharSpace; // v7: character space WORD m_wReserved2; BYTE m_nLineSpacePrint; // space between lines BYTE m_byteReserved1; WORD m_wReserved1; int m_nReserved15; // was m_nHiliteTag before v3.16 int m_nReserved14; // was m_nHiliteMultiComment before v3.16 UINT m_nAutoSaveTime; // auto save time int m_nCheckFileChanged; // v3: changed by another program UINT m_nUndoBufferSize; // undo max number int m_nEncodingNew; // v3: encoding for new files int m_nCrLfNew; // v3: how to return for new files int m_nCharsetNew; // v3: font charset for new files int m_nEncodingWrite; // v3: encoding for saving int m_nCrLfWrite; // v3: how to return for saving int m_nSpecialSyntax; // v3.16: Special Syntax WCHAR m_chEscape; // v3.16: Escape character bool m_bPasteAnsi; // v3.16: Always Paste as ANSI bool m_bNewTemplate; // v3.17: Use template for a new file bool m_bSaveAsNameEntity; // v3.17: Use Named Entity Reference bool m_bInsertSpacesTab; // v3.19: Insert spaces for Tab WCHAR m_chIndentBegin; // v3.17: Begin Indent WCHAR m_chIndentEnd; // v3.17: End Indent WCHAR m_chEndOfStatement; // v3.17: End of Statement int m_nIndentSpace; // v3.19: Indent columns bool m_bNoSpaceEdge; // v3.19: No space at left edge of Window bool m_bAnsiFont; // v3.28: Non-Unicode Font --- obsolete bool m_bShowScrollOnlyActive; // v3.31: Show scroll bars only when current pane is active bool m_bWrapPagePrint; // v3.31: Wrap by Page when printing int m_nPrinterMarginRight; // v3.24: printer right margin int m_nMaxFindHilite; // v3.32: (Depth of searched string to highlight) - 1 bool m_bPromptInvalidChar; // v4.01: Prompt if invalid characters found bool m_bNameUntitled; // 6.00: auto name untitled -- v4.03: Synchronize Read-Only attribute BYTE m_byteMonitorInterval; // 5.00: monitor interval for changed file (File tab) bool m_bVirtualSpace; // 7.00 : Enable Virtual Space // m_bReserved1; BYTE m_byteSmoothScrollSpeed; // 7.00 : Smooth Scroll Speed MIN_SMOOTH_SCROLL_SPEED (slow) - MAX_SMOOTH_SCROLL_SPEED (fast) bool m_bReserved3; // m_bRightAllBeyond; // 7.00 : Right All Beyond bool m_bReserved2; bool m_bReserved1; int m_nReserved5; // reserved int m_nReserved4; // reserved int m_nReserved3; // reserved int m_nReserved2; // reserved int m_nReserved1; // reserved BYTE m_abUrlChar[128]; // =1: URL char, =2: URL char byte not at end. bool m_bNotepadDiary; // notepad compatible diary bool m_bPrintLineNum; // print line numbers bool m_bPromptNullFile; // prompt if Null character found bool m_bPromptCrLf; // prompt at inconsistent returns bool m_bShowEOF; // show EOF bool m_bShowCR; // show returns bool m_bShowTab; // show tab bool m_bShowLineNum; // show line numbers BYTE m_bShowLogicalLine; // line and column display as bool m_bWordWrap; // word wrap bool m_bFaceWrap; // enable non-wrap words bool m_bKinsokuWrap; // wrap these characters bool m_bSaveTabToSpace; // save tabs as spaces bool m_bSaveInsertCR; // insert returns when saving bool m_bUseRecycleBin; // use recylce bin to buckup bool m_bAutoIndent; // auto indent bool m_bWrapIndent; // v7.00: auto indent for wrap position, used to be m_bOverwrite, v6.00 Obsolete bool m_bHilite; // highlight these words bool m_bURL; // link to URLs bool m_bMailTo; // clicking mail address sends mail bool m_bLinkDblclick; // enable double clicking only bool m_bFullPath; // show file name with full path bool m_b7BitKanji; // 7 bit kanji bool m_bCrLfSeparateMark; // show CR and LF with different marks bool m_bShowRuler; // show ruler bool m_bAutoSave; // auto save bool m_bDeleteEmpty; // delete empty files when saving bool m_bSaveNotModified; // always enable saving bool m_bBackupFolder; // save backups to backup folder bool m_bFolderIfRecycleFailed; // save to backup folder if recycle bin not available bool m_bAutoSaveFolder; // save to auto save folder bool m_bRenameBackup; // rename if same file name exists bool m_bControlIME; // run input method editor bool m_bRenameAutoSave; // rename if same file name exists bool m_bBackupSame; // save backups to same folder bool m_bShowDbSpace; // show double-byte spaces bool m_bSelLinkContextMenu; // v3: not implemented bool m_bPageVScroll; // v3: always enable 1page vertical scroll bool m_bPageHScroll; // v3: always enable 1page horizontal scroll bool m_bHalfPageScroll; // v3: scroll half page bool m_bDetectUnicode; // v3: detect Unicode(UTF-16/UTF-8)signature (BOM) bool m_bAllowCtrlChars; // v3: allow insert control character bool m_bMoveCursorScroll; // v3: move cursor by scrolling bool m_bHorzLine; // v3: horizontal line bool m_bVertLine; // v3: vertical line bool m_bScroll2Lines; // v3: double line scroll bool m_bFastKeyRepeat; // v3: faster cursor movement bool m_bDBCharUrl; // v3: recognize double-byte characters as URLs bool m_bKanaUrl; // v3: recognize single-byte kana and kana marks as URLs bool m_bShowPage; // v3: display page number bool m_bUsePrinterFont; // v3: choose font for default printer bool m_bSignatureNew; // v3: Unicode, UTF-8 signature for new files bool m_bPromptNotAnsi; // v3: prompt on saving if unicode characters cannot convert to ANSI bool m_bSignatureWrite; // v3: Unicode, UTF-8 signature for saving bool m_bIgnoreColorPrint; // v3.08: Ignore Color and Underlines (Print) bool m_bNoFullPathIfNotActive; // v3.08: Display file name without full path if the window is not active bool m_bSmoothScroll; // v7.00: Smooth Scroll, used to be m_bSaveOverwrite, v6.00 Obsolete bool m_bNoChangeCrLf; // v3.13: Do not change how to return at copy and paste. bool m_bShowSpace; // v3.13: show single-byte spaces bool m_bWordWrapMark; // v3.13: allows word wrap after marks bool m_bPrintSeparatingLines; // v3.14: Draw separating lines bool m_bSameFontPrint; // v3.14: Use Display Font as Printer Font bool m_bHiliteCorresParen; // v3.14: Highlight Corresponding Parentheses bool m_bSelectInQuotes; // v3.14: Highlight and easily select in "quotes". bool m_bDetectUTF8; // v3.15: Detect UTF-8 bool m_bDetectCharset; // v3.15: Detect Charset (HTML) bool m_bDetectAll; // v3.15: Detect All bool m_bDeleteSpaceEnd; // v3.15: Delete Space at End of Lines bool m_bSaveAsEntity; // v3.15: Encode Unicode as HTML Entity bool m_bShowControl; // v3.15: Highlight Control Characters BYTE m_bQuoteType; // v3.16: Quote type, combination of QUOTE_SINGLE, QUOTE_DOUBLE and QUOTE_CONTINUE WCHAR m_chKanjiInChar; // transitional character to kanji WCHAR m_chKanjiOutChar; // transitional character to single-bytes WCHAR m_chTagLeft; // begin tag WCHAR m_chTagRight; // end tag WCHAR m_szHeader [MAX_HEADER]; // header WCHAR m_szLineComment1[MAX_LINE_COMMENT]; // v3.16: Line Comment WCHAR m_szScriptBegin [MAX_SCRIPT_BEGIN]; // v3.16: Script Begin WCHAR m_szFooter[MAX_FOOTER]; // footer WCHAR m_szLineComment2 [MAX_LINE_COMMENT]; // v3.16: Line Comment WCHAR m_szScriptEnd[MAX_SCRIPT_END]; // v3.16: Script End WCHAR m_szPrefix [MAX_PREFIX_LENGTH]; // default quote mark WCHAR m_szKinsokuBegin[MAX_KINSOKU_BEGIN]; // not allowed at line head WCHAR m_szKinsokuEnd[MAX_KINSOKU_END]; // not allowed at line end WCHAR m_szDefExt[MAX_DEF_EXT]; // default extension WCHAR m_szPrefixList[MAX_PREFIX_LIST]; // quote mark WCHAR m_szBackupPath[MAX_PATH]; // backup folder WCHAR m_szAutoSavePath[MAX_PATH]; // auto save folder WCHAR m_szMultiCommentBegin[MAX_MULTI_COMMENT_BEGIN]; // Multi-line comment begin WCHAR m_szMultiCommentEnd[MAX_MULTI_COMMENT_END]; // Multi-line comment end public: void Initialize(); }; // Command IDs // #define EEID_FILE_NEW 4096 #define EEID_FILE_OPEN 4097 #define EEID_FILE_CLOSE_OPEN 4098 #define EEID_FILE_SAVE 4099 #define EEID_FILE_SAVE_AS 4100 #define EEID_FILE_SAVE_ALL 4101 #define EEID_FILE_SAVE_ANSI 4102 #define EEID_FILE_SAVE_JIS 4103 #define EEID_FILE_SAVE_EUC 4104 #define EEID_SAVE_AS_CRLF 4105 #define EEID_SAVE_AS_CR 4106 #define EEID_SAVE_AS_LF 4107 #define EEID_FILE_INSERT 4108 #define EEID_FILE_RELOAD 4109 #define EEID_FILE_RELOAD_ANSI 4110 #define EEID_FILE_RELOAD_JIS 4111 #define EEID_FILE_RELOAD_EUC 4112 #define EEID_READ_ONLY 4113 #define EEID_FILE_PRINT 4114 #define EEID_FILE_PRINT_DIRECT 4115 #define EEID_FILE_SAVE_EXIT 4116 #define EEID_APP_EXIT 4117 #define EEID_SAVE_EXIT_ALL 4118 #define EEID_EXIT_ALL 4119 #define EEID_APP_QUIT 4120 #define EEID_NEW_TRAY_ICON 4121 #define EEID_TRAY_ICON_EXIT 4122 #define EEID_FILE_NEW_PASTE 4123 #define EEID_EDIT_UNDO 4124 #define EEID_EDIT_REDO 4125 #define EEID_EDIT_CUT 4126 #define EEID_EDIT_COPY 4127 #define EEID_EDIT_COPY_DESELECT 4128 #define EEID_EDIT_PASTE 4129 #define EEID_EDIT_COPY_PREFIX 4130 #define EEID_EDIT_COPY_PREFIX_DESELECT 4131 #define EEID_PASTE_PREFIX 4132 #define EEID_PASTE_RETURN 4133 #define EEID_PASTE_PREFIX_RETURN 4134 #define EEID_DELETE 4135 #define EEID_EDIT_SELECT_ALL 4136 #define EEID_INSERT_DATE 4137 #define EEID_INSERT_DATE_TIME 4138 #define EEID_JUMP 4139 #define EEID_EDIT_COPY_LINK 4140 #define EEID_LINK_OPEN 4141 #define EEID_INSERT 4142 #define EEID_INSERT_CR_WRAP 4143 #define EEID_DELETE_CR_WRAP 4144 #define EEID_INSERT_CR 4145 #define EEID_INSERT_LF 4146 #define EEID_TAG_JUMP 4147 #define EEID_CONVERT 4148 #define EEID_MAKE_UPPER 4149 #define EEID_MAKE_LOWER 4150 #define EEID_ZEN_TO_HAN 4151 #define EEID_HAN_TO_ZEN

4152 #define EEID_SELECT_CHAR 4153 #define EEID_SELECT_LINE 4154 #define EEID_SELECT_BOX 4155 #define EEID_RIGHT 4156 #define EEID_LEFT 4157 #define EEID_RIGHT_WORD 4158 #define EEID_LEFT_WORD 4159 #define EEID_UP 4160 #define EEID_DOWN 4161 #define EEID_PAGEUP 4162 #define EEID_PAGEDOWN 4163 #define EEID_HOME 4164 #define EEID_LOGICAL_HOME 4165 #define EEID_END 4166 #define EEID_LOGICAL_END 4167 #define EEID_TOP 4168 #define EEID_BOTTOM 4169 #define EEID_SCROLL_UP 4170 #define EEID_SCROLL_DOWN 4171 #define EEID_SHIFT_RIGHT 4172 #define EEID_SHIFT_LEFT 4173 #define EEID_SHIFT_RIGHT_WORD 4174 #define EEID_SHIFT_LEFT_WORD 4175 #define EEID_SHIFT_UP 4176 #define EEID_SHIFT_DOWN 4177 #define EEID_SHIFT_PAGEUP 4178 #define EEID_SHIFT_PAGEDOWN 4179 #define EEID_SHIFT_HOME 4180 #define EEID_SHIFT_LOGICAL_HOME 4181 #define EEID_SHIFT_END 4182 #define EEID_SHIFT_LOGICAL_END 4183 #define EEID_SHIFT_TOP 4184 #define EEID_SHIFT_BOTTOM 4185 #define EEID_BACK 4186 #define EEID_ESCAPE 4187 #define EEID_TAB 4188 #define EEID_SHIFT_TAB 4189 #define EEID_DELETE_LINE 4190 #define EEID_DELETE_RIGHT_LINE 4191 #define EEID_EDIT_COPY_LINE 4192 #define EEID_EDIT_CUT_LINE 4193 #define EEID_DELETE_WORD 4194 #define EEID_LINE_OPEN_ABOVE 4195 #define EEID_LINE_OPEN_BELOW 4196 #define EEID_INSERT_CONTROL 4197 #define EEID_TOGGLE_IME 4198 #define EEID_RECONVERT 4199 #define EEID_EDIT_FIND 4200 #define EEID_EDIT_REPLACE 4201 #define EEID_EDIT_REPEAT 4202 #define EEID_EDIT_REPEAT_BACK 4203 #define EEID_FIND_NEXT_WORD 4204 #define EEID_FIND_PREV_WORD 4205 #define EEID_ERASE_FIND_HILITE 4206 #define EEID_GREP 4207 #define EEID_WRAP_NONE 4208 #define EEID_WRAP_BY_CHAR 4209 #define EEID_WRAP_BY_WINDOW 4210 #define EEID_VIEW_STATUS_BAR 4212 #define EEID_WATCH_CHAR_CODE 4213 #define EEID_NEXT_PANE 4214 #define EEID_PREV_PANE 4215 #define EEID_QUICK_MACRO_RECORD 4216 #define EEID_QUICK_MACRO_RUN 4217 #define EEID_FONT 4218 #define EEID_CUSTOMIZE 4219 #define EEID_CONFIG_POPUP 4220 #define EEID_CONFIG 4221 #define EEID_COMMON_SETTINGS 4222 #define EEID_FILE_ASSOCIATE 4223 #define EEID_CUSTOMIZE_TOOLBAR 4224 #define EEID_CUSTOMIZE_PLUG_INS 4238 #define EEID_WINDOW_ALWAYS_TOP 4239 #define EEID_WINDOW_SPLIT 4240 #define EEID_WINDOW_CASCADE 4241 #define EEID_WINDOW_TILE_HORZ 4242 #define EEID_WINDOW_TILE_VERT 4243 #define EEID_WINDOW_MINIMIZE_ALL 4244 #define EEID_NEXT_WINDOW 4245 #define EEID_PREV_WINDOW 4246 #define EEID_HELP_FINDER 4247 #define EEID_HELP_REGIST 4248 #define EEID_WEB_HOME 4249 #define EEID_APP_ABOUT 4250 #define EEID_SELECT_WORD 4251 #define EEID_FILE_SAVE_RENAME 4252 #define EEID_TAB_TO_SPACE 4253 #define EEID_FILE_SAVE_UNICODE 4254 #define EEID_FILE_SAVE_UTF8 4255 #define EEID_FILE_SAVE_UTF7 4256 #define EEID_FILE_RELOAD_UNICODE 4257 #define EEID_FILE_RELOAD_UTF8 4258 #define EEID_FILE_RELOAD_UTF7 4259 #define EEID_FILE_SAVE_UNICODE_BIGENDIAN 4260 #define EEID_FILE_RELOAD_UNICODE_BIGENDIAN 4261 #define EEID_EDIT_PASTE_ANSI 4262 #define EEID_FILE_RELOAD_932 4263 #define EEID_DEFINE_CODE_PAGE 4264 #define EEID_FILE_SAVE_932 4265 #define EEID_CUSTOMIZE_MENU 4266 #define EEID_ALL_COMMANDS 4267 #define EEID_NEXT_LOGICAL_LINE 4268 #define EEID_PREV_LOGICAL_LINE 4269 #define EEID_ALL_PROP 4270 #define EEID_NEW_PASTE_PREFIX 4271 #define EEID_NEW_PASTE_PREFIX_RETURN 4272 #define EEID_CUSTOMIZE_TRAY 4273 #define EEID_INSERT_CR_LF 4274 #define EEID_DELETE_RIGHT_WORD 4275 #define EEID_NEXT_PAREN 4276 #define EEID_SHIFT_NEXT_PAREN 4277 #define EEID_TRIM_RIGHT 4278 #define EEID_FILE_RELOAD_DETECT_ALL 4279 #define EEID_DELETE_LEFT_WORD 4280 #define EEID_NEW_CONFIG_POPUP 4281 #define EEID_FONT_POPUP 4282 #define EEID_RELOAD_POPUP 4283 #define EEID_DELETE_PANES 4284 #define EEID_SHOW_PLUGINS_BAR 4285 #define EEID_PRINT_PREVIEW 4286 #define EEID_WINDOW_TOP 4292 #define EEID_WINDOW_BOTTOM 4293 #define EEID_WINDOW_RIGHT 4294 #define EEID_WINDOW_LEFT 4295 #define EEID_HOME_TEXT 4296 #define EEID_SHIFT_HOME_TEXT 4297 #define EEID_KEYBOARD_MAP 4298 #define EEID_WINDOW_SPLIT_HORZ 4299 #define EEID_WINDOW_SPLIT_VERT 4300 #define EEID_CONTEXT_MENU 4301 #define EEID_DELETE_LEFT_LINE 4302 #define EEID_INSERT_GRAVE 4303 #define EEID_INSERT_ACUTE 4304 #define EEID_INSERT_CIRCUMFLEX 4305 #define EEID_INSERT_TILDE 4306 #define EEID_INSERT_DIAERESIS 4307 #define EEID_INSERT_RING_ABOVE 4308 #define EEID_INSERT_LIGATURE 4309 #define EEID_INSERT_CEDILLA 4310 #define EEID_INSERT_STROKE 4311 #define EEID_INSERT_INVERTED_QUESTION 4312 #define EEID_INSERT_INVERTED_EXCLAMATION 4313 #define EEID_INSERT_COPYRIGHT 4314 #define EEID_INSERT_REGISTERED 4315 #define EEID_INSERT_TRADEMARK 4316 #define EEID_INSERT_EURO 4317 #define EEID_WRAP_BY_PAPER 4318 #define EEID_SHOW_TOOLS_BAR 4319 #define EEID_BOOKMARK_TOGGLE 4320 #define EEID_BOOKMARK_NEXT 4321 #define EEID_BOOKMARK_PREV 4322 #define EEID_BOOKMARK_CLEAR 4323 #define EEID_CUSTOMIZE_TOOLS 4324 #define EEID_RETRIEVE_FIND_TEXT 4325 #define EEID_COPY_FILE_PATH 4326 #define EEID_COPY_FILE_DIR 4327 #define EEID_DUPLICATE_LINE 4328 #define EEID_LOAD_WORKSPACE 4329 #define EEID_SAVE_WORKSPACE 4330 #define EEID_SAVE_WORKSPACE_EXIT_ALL 4331 #define EEID_SAVE_WORKSPACE_QUIT_ALL 4332 #define EEID_LOGICAL_HOME_TEXT 4333 #define EEID_SHIFT_LOGICAL_HOME_TEXT 4334 #define EEID_WINDOW_SPLIT_HORZ_FIX 4335 #define EEID_WINDOW_SPLIT_VERT_FIX 4336 //#define EEID_SHOW_WINDOWS_BAR 4337 #define EEID_SHOW_BAR_TITLE 4340 #define EEID_LOCK_TOOLBARS 4341 #define EEID_WINDOW_COMBINE 4342 #define EEID_WINDOW_ALWAYS_TOP_ON 4343 #define EEID_WINDOW_ALWAYS_TOP_OFF 4344 #define EEID_MOVE_LAST_EDIT 4345 #define EEID_MACRO_SAVE 4346 #define EEID_MACRO_SELECT 4347 #define EEID_MACRO_EDIT 4348 #define EEID_MACRO_SELECT_THIS 4349 #define EEID_CUSTOMIZE_MACRO 4350 #define EEID_BOOKMARK_NEXT_WITHIN 4351 #define EEID_BOOKMARK_PREV_WITHIN 4352 #define EEID_BOOKMARK_SET 4353 #define EEID_BOOKMARK_RESET 4354 #define EEID_SPACE_TO_TAB 4355 #define EEID_TABIFY 4356 #define EEID_UNTABIFY 4357 #define EEID_INDENT 4358 #define EEID_UNINDENT 4359 #define EEID_MACRO_HELP 4360 #define EEID_MACRO_HELP_WORD 4361 #define EEID_REPLACE_IN_FILES 4362 #define EEID_QUIT_ALL 4363 #define EEID_MACRO_RUN_OPTIONS 4364 #define EEID_INSERT_CARON 4369 #define EEID_VIEW_MARKS 4370 #define EEID_EDIT_COMMENT 4371 #define EEID_EDIT_UNCOMMENT 4372 #define EEID_INCREASE_FONT_SIZE 4373 #define EEID_DECREASE_FONT_SIZE 4374 #define EEID_FIND_NEXT_UNICODE 4375 #define EEID_FIND_PREV_UNICODE 4376 #define EEID_ERASE_UNICODE_HILITE 4377 #define EEID_JOIN_LINES 4378 #define EEID_SPLIT_LINES 4379 #define EEID_IMPORT_EXPORT 4380 #define EEID_CAPITALIZE 4381 #define EEID_WINDOW_MOVE_NEXT 4382 #define EEID_WINDOW_MOVE_PREV 4383 #define EEID_CLOSE_ALL_OTHERS 4384 #define EEID_WINDOW_SPLIT_HORZ_TOGGLE 4385 #define EEID_WINDOW_SPLIT_VERT_TOGGLE 4386 #define EEID_GROUP_CLOSE_ALL 4387 #define EEID_GROUP_CLOSE_OTHERS 4388 #define EEID_GROUP_CLOSE_LEFT 4389 #define EEID_GROUP_CLOSE_RIGHT 4390 #define EEID_NEW_GROUP 4391 #define EEID_NEW_GROUP_MINIMIZE 4392 #define EEID_NEW_GROUP_CASCADE 4393 #define EEID_NEW_GROUP_HORZ 4394 #define EEID_NEW_GROUP_VERT 4395 #define EEID_MOVE_PREV_GROUP 4396 #define EEID_MOVE_NEXT_GROUP 4397 #define EEID_SORT_FILE_NAME 4398 #define EEID_SORT_TYPE 4399 #define EEID_SORT_MODIFIED 4400 #define EEID_SORT_ZORDER 4401 #define EEID_SORT_ASCENDING 4402 #define EEID_SORT_DESCENDING 4403 #define EEID_AUTO_SORT 4404 #define EEID_RESTORE_POS 4406 #define EEID_WINDOW_COMBINE_ON 4407 #define EEID_WINDOW_COMBINE_OFF 4408 #define EEID_ACTIVE_PANE 4409 #define EEID_OUTLINE_COLLAPSE_ALL 4410 #define EEID_OUTLINE_EXPAND_ALL 4411 #define EEID_OUTLINE_TOGGLE_LINE 4412 #define EEID_OUTLINE_COLLAPSE_LINE 4413 #define EEID_OUTLINE_EXPAND_LINE 4414 #define EEID_OUTLINE_NEXT_NODE 4415 #define EEID_OUTLINE_PREV_NODE 4416 #define EEID_SHIFT_NEXT_NODE 4417 #define EEID_SHIFT_PREV_NODE 4418 #define EEID_RESTORE_DELETED 4419 #define EEID_VIEW_OUTPUT 4420 #define EEID_SHOW_MACROS_BAR 4421 #define EEID_REFRESH_TOOLBARS 4422 #define EEID_RECORD_MOUSE 4423 #define EEID_RECORD_NO_MOUSE 4424 #define EEID_FOCUS_LEFT_BAR 4425 #define EEID_FOCUS_TOP_BAR 4426 #define EEID_FOCUS_RIGHT_BAR 4427 #define EEID_FOCUS_BOTTOM_BAR 4428 #define EEID_USER_MENU1 4429 #define EEID_USER_MENU2 4430 #define EEID_USER_MENU3 4431 #define EEID_USER_MENU4 4432 #define EEID_USER_MENU5 4433 #define EEID_USER_MENU6 4434 #define EEID_USER_MENU7 4435 #define EEID_USER_MENU8 4436 #define EEID_SELECT_LOGICAL_LINE 4437 #define

```
EEID_FILE_MRU_FILE1 4609 // to EEID_FILE_MRU_FILE1 + 63 #define EEID_MRU_FONT1 4736 // to EEID_MRU_FONT1 + 63 #define
EEID_FILE_MRU_INSERT1 4864 // to EEID_FILE_MRU_INSER1 + 63 #define EEID_FILE_MRU_FOLDER1 4992 // to
EEID_FILE_MRU_FOLDER1 + 63 #define EEID_SELECT_CONFIG 5120 // to EEID_SELECT_CONFIG + 255 #define
EEID_WINDOW_MENU 5376 // to EEID_WINDOW_MENU + 255 #define EEID_PLUG_IN1 5632 // to EEID_PLUG_IN1 + 255 #define
EEID_FILE_RELOAD_DEFINED 6656 // to EEID_FILE_RELOAD_DEFINED + 255 #define EEID_FILE_NEW_CONFIG 7168 // to
EEID_FILE_NEW_CONFIG + 255 #define EEID_FILE_SAVE_DEFINED 7680 // to EEID_FILE_SAVE_DEFINED + 255 #define
EEID_TOOL1 8192 // to EEID_TOOL1 + 255 #define EEID_MACRO1 9216 // to EEID_MACRO1 + 1023 #define EEID_CUSTOM_REBAR1
21504 // to EEID_CUSTOM_REBAR1 + 255 #define EEID_CHARSET_DEFAULT 8704 #define EEID_CHARSET_ARABIC 8705 #define
EEID_CHARSET_BALTIC 8706 #define EEID_CHARSET_CENTRAL_EUROPE 8707 #define EEID_CHARSET_CHINESE_SIMPLIFIED
8708 #define EEID_CHARSET_CHINESE_TRADITIONAL 8709 #define EEID_CHARSET_CYRILLIC 8710 #define
EEID_CHARSET_GREEK 8711 #define EEID_CHARSET_HEBREW 8712 #define EEID_CHARSET_JAPANESE 8713 #define
EEID_CHARSET_KOREAN 8714 #define EEID_CHARSET_THAI 8715 #define EEID_CHARSET_TURKISH 8716 #define
EEID_CHARSET_VIETNAMESE 8717 #define EEID_CHARSET_WESTERN_EUROPE 8718 #define EEID_CHARSET_OEM 8719 #define
EEID_PROPERTY_MARGIN 8960 #define EEID_PROPERTY_SCROLL 8961 #define EEID_PROPERTY_FILE 8962 #define
EEID_PROPERTY_BACKUP 8963 #define EEID_PROPERTY_AUTO_SAVE 8964 #define EEID_PROPERTY_ASSOCIATE 8965 #define
EEID_PROPERTY_KINSOKU 8966 #define EEID_PROPERTY_FACE 8967 #define EEID_PROPERTY_HILITE 8968 #define
EEID_PROPERTY_COMMENT 8969 #define EEID_PROPERTY_SHOW 8970 #define EEID_PROPERTY_MARK 8971 #define
EEID_PROPERTY_PRINT 8972 #define EEID_PROPERTY_LINK 8973 #define EEID_PROPERTY_KEYBOARD 8974 #define
EEID_CUSTOMIZE_FILE 9040 #define EEID_CUSTOMIZE_SEARCH 9041 #define EEID_CUSTOMIZE_HISTORY 9042 #define
EEID_CUSTOMIZE_WINDOW 9043 #define EEID_CUSTOMIZE_TAB 9044 #define EEID_CUSTOMIZE_STATUS 9045 #define
EEID_CUSTOMIZE_ADVANCED 9046 #if _MSC_VER > 1000 #pragma once #endif #include "plugin.h" #include #pragma warning( push )
#pragma warning( disable : 4995 ) // 'function': name was marked as #pragma deprecated #if _MSC_VER < 0x0700 #pragma warning( disable :
4786 ) // dentifier was truncated to '255' characters in the browser information #pragma warning( disable : 4512 ) // assignment operator could
not be generated #pragma warning( disable : 4100 ) // unreferenced formal parameter #endif #include #pragma warning( pop ) #if _MSC_VER
< 0x0700 #pragma warning( push ) #pragma warning( disable : 4127 ) // conditional expression is constant #pragma warning( disable : 4786 ) //
dentifier was truncated to '255' characters in the browser information #endif #ifndef _countof #define _countof(array) (sizeof(array)/sizeof
(array[0])) #endif // these asserts fail on previous versions of EmEditor (v4.13 or earlier). #ifdef TEST_V5 #define ASSERT_STRICT(x)
_ASSERTE(x) #else #define ASSERT_STRICT(x) (void(0)) #endif #ifdef EE_IMPLEMENT #define EE_EXTERN #else #define EE_EXTERN
extern #endif #ifdef _DEBUG #ifdef _DEBUG_NEW #define new _DEBUG_NEW #endif #endif // forward declaration #define
DEFINE_CREATE(c) class ETL_FRAME_CLASS_NAME; template class CETLFrame; typedef CETLFrame CETLFrameX; typedef
std::map CETLFrameMap; CETLFrameX* _ETLCreateFrame(); void _ETLDeleteFrame( CETLFrameX* pFrame ); #define
_ETL_IMPLEMENT CETLFrameX* _ETLCreateFrame() { CETLFrameX* pFrame = new ETL_FRAME_CLASS_NAME; return pFrame; }
void _ETLDeleteFrame( CETLFrameX* pFrame ) { delete static_cast(pFrame); } // global data definition class CETLData { public:
HINSTANCE m_hInstance; CRITICAL_SECTION m_cs; CETLFrameMap* m_pETLFrameMap; WORD m_wCmdID; CETLData()
{ ZeroMemory( this, sizeof( CETLData ) ); InitializeCriticalSection( &m_cs ); } ~CETLData() { DeleteCriticalSection( &m_cs ); } void Lock()
{ EnterCriticalSection( &m_cs ); } void Unlock() { ::LeaveCriticalSection( &m_cs ); } } _ETLData; class CETLLock { public: CETLLock()
{ _ETLData.Lock(); } ~CETLLock() { _ETLData.Unlock(); } }; template class __declspec(novtable) CETLFrame { public: HWND m_hWnd;
CETLFrame() : m_hWnd(0) { } ~CETLFrame() { } virtual void OnCommand( HWND hwndView ) { T* pT = static_cast(this); pT-
>OnCommand( hwndView ); } void OnEvents( HWND hwndView, UINT nEvent, LPARAM lParam ) { T* pT = static_cast(this); pT-
>OnEvents( hwndView, nEvent, lParam ); } BOOL QueryStatus( HWND hwndView, LPBOOL pbChecked ) { T* pT = static_cast(this); return
pT->QueryStatus( hwndView, pbChecked ); } static UINT GetMenuTextID() { return T::_IDS_MENU; } static UINT GetStatusMessageID()
{ return T::_IDS_STATUS; } static UINT GetBitmapID() { return T::_IDB_BITMAP; } static LRESULT GetStringA( LPSTR pBuf, size_t
cchBuf, UINT nID ) { char sz[80]; LRESULT lResult = LoadStringA( EEGetInstanceHandle(), nID, sz, _countof( sz ) ) + 1; _ASSERTE( lResult
> 1 ); if( pBuf ){ lstrcpynA( pBuf, sz, (int)cchBuf ); } return lResult; } static LRESULT GetStringW( LPWSTR pBuf, size_t cchBuf, UINT nID )
{ WCHAR sz[80]; LRESULT lResult = LoadStringW( EEGetInstanceHandle(), nID, sz, _countof( sz ) ) + 1; // LoadStringW does not work on
Windows 9x, but that is OK. _ASSERTE( lResult > 1 ); if( pBuf ){ lstrcpynW( pBuf, sz, (int)cchBuf ); } return lResult; } static LRESULT
GetNameA( LPSTR pBuf, size_t cchBuf ) { return GetStringA( pBuf, cchBuf, T::_IDS_NAME ); } static LRESULT GetNameW( LPWSTR
pBuf, size_t cchBuf ) { return GetStringW( pBuf, cchBuf, T::_IDS_NAME ); } static LRESULT GetVersionA( LPSTR pBuf, size_t cchBuf )
{ return GetStringA( pBuf, cchBuf, T::_IDS_VER ); } static LRESULT GetVersionW( LPWSTR pBuf, size_t cchBuf ) { return GetStringW
( pBuf, cchBuf, T::_IDS_VER ); } static LRESULT GetInfo( WPARAM flag ) { LRESULT lResult = 0; switch( flag ){ case
EPGI_ALLOW_OPEN_SAME_GROUP: lResult = T::_ALLOW_OPEN_SAME_GROUP; break; case
EPGI_ALLOW_MULTIPLE_INSTANCES: lResult = T::_ALLOW_MULTIPLE_INSTANCES; break; case EPGI_MAX_EE_VERSION:
lResult = T::_MAX_EE_VERSION; break; case EPGI_MIN_EE_VERSION: lResult = T::_MIN_EE_VERSION; break; case
EPGI_SUPPORT_EE_PRO: lResult = T::_SUPPORT_EE_PRO; break; case EPGI_SUPPORT_EE_STD: lResult = T::_SUPPORT_EE_STD;
break; } return lResult; } LRESULT PlugInProc( HWND hwnd, UINT nMsg, WPARAM wParam, LPARAM lParam ) { LRESULT lResult = 0;
T* pT = static_cast(this); switch( nMsg ){ case EP_QUERY_UNINSTALL: lResult = pT->QueryUninstall( hwnd ); break; case
EP_SET_UNINSTALL: lResult = pT->SetUninstall( hwnd, (LPTSTR)wParam, (LPTSTR)lParam ); break; case EP_QUERY_PROPERTIES:
lResult = pT->QueryProperties( hwnd ); break; case EP_SET_PROPERTIES: lResult = pT->SetProperties( hwnd ); break; case
EP_PRE_TRANSLATE_MSG: lResult = pT->PreTranslateMessage( hwnd, (MSG*)lParam ); break; } return lResult; } static LRESULT
GetMask( WPARAM wParam ) { LRESULT lResult = 0; if( wParam & BITMAP_24BIT_COLOR ){ lResult = T::_MASK_TRUE_COLOR; }
else if( wParam & BITMAP_256_COLOR ){ lResult = T::_MASK_256_COLOR; } return lResult; } static LRESULT GetBitmap( WPARAM
wParam ) { LRESULT lResult = 0; if( wParam & BITMAP_LARGE ){ if( wParam & BITMAP_24BIT_COLOR ){ if( wParam &
BITMAP_DISABLED ){ lResult = T::_IDB_TRUE_24_BW; } else if( wParam & BITMAP_HOT ){ lResult = T::_IDB_TRUE_24_HOT; } else
{ lResult = T::_IDB_TRUE_24_DEFAULT; } } else if( wParam & BITMAP_256_COLOR ){ if( wParam & BITMAP_DISABLED ){ lResult =
T::_IDB_256C_24_BW; } else if( wParam & BITMAP_HOT ){ lResult = T::_IDB_256C_24_HOT; } else { lResult =
T::_IDB_256C_24_DEFAULT; } } else { lResult = T::_IDB_16C_24; } } else { if( wParam & BITMAP_24BIT_COLOR ){ if( wParam &
BITMAP_DISABLED ){ lResult = T::_IDB_TRUE_16_BW; } else if( wParam & BITMAP_HOT ){ lResult = T::_IDB_TRUE_16_HOT; } else
{ lResult = T::_IDB_TRUE_16_DEFAULT; } } else if( wParam & BITMAP_256_COLOR ){ if( wParam & BITMAP_DISABLED ){ lResult =
T::_IDB_256C_16_BW; } else if( wParam & BITMAP_HOT ){ lResult = T::_IDB_256C_16_HOT; } else { lResult =
T::_IDB_256C_16_DEFAULT; } } else { lResult = T::_IDB_BITMAP; } } return lResult; } // Registry/INI void GetProfileString( DWORD
dwKey, LPCWSTR pszConfig, LPCWSTR lpszEntry, LPWSTR lpszBuf, DWORD cchBufSize, LPCWSTR lpszDefault ) { DWORD dwSize =
cchBufSize * sizeof( WCHAR ); if( ERROR_SUCCESS == Editor_RegQueryValue( m_hWnd, dwKey, pszConfig, lpszEntry, REG_SZ,
(LPBYTE)lpszBuf, &dwSize, 0 ) ){ return; } lstrcpynW( lpszBuf, lpszDefault, cchBufSize ); } void GetProfileString( LPCWSTR lpszEntry,
LPWSTR lpszBuf, DWORD cchBufSize, LPCWSTR lpszDefault ) { TCHAR szFileName[MAX_PATH]; if( GetModuleFile( szFileName ) )
{ GetProfileString( EEREG_EMEDITORPLUGIN, szFileName, lpszEntry, lpszBuf, cchBufSize, lpszDefault ); return; } lstrcpynW( lpszBuf,
lpszDefault, cchBufSize ); } int GetProfileInt( DWORD dwKey, LPCWSTR pszConfig, LPCWSTR lpszEntry, int nDefault ) { _ASSERT
( lpszEntry ); DWORD dwSize = sizeof( DWORD ); DWORD dwData = 0; if( ERROR_SUCCESS == Editor_RegQueryValue( m_hWnd,
dwKey, pszConfig, lpszEntry, REG_DWORD, (LPBYTE)&dwData, &dwSize, 0 ) ){ return (int)dwData; } return nDefault; } int GetProfileInt
( LPCWSTR lpszEntry, int nDefault ) { TCHAR szFileName[MAX_PATH]; if( GetModuleFile( szFileName ) ) { return GetProfileInt
( EEREG_EMEDITORPLUGIN, szFileName, lpszEntry, nDefault ); } return nDefault; } DWORD GetProfileBinary( DWORD dwKey,
LPCWSTR pszConfig, LPCWSTR lpszEntry, LPBYTE lpBuf, DWORD cbBufSize ) { _ASSERT( lpszEntry ); if( ERROR_SUCCESS ==
Editor_RegQueryValue( m_hWnd, dwKey, pszConfig, lpszEntry, REG_BINARY, (LPBYTE)lpBuf, &cbBufSize, 0 ) ){ return cbBufSize; }
return 0; } DWORD GetProfileBinary( LPCWSTR lpszEntry, LPBYTE lpBuf, DWORD cbBufSize ) { TCHAR szFileName[MAX_PATH]; if
```

```
( GetModuleFile( szFileName ) ) { return GetProfileBinary( EEREG_EMEDITORPLUGIN, szFileName, lpszEntry, lpBuf, cbBufSize ); } return
0; } void WriteProfileString( DWORD dwKey, LPCWSTR pszConfig, LPCWSTR lpszEntry, LPCWSTR lpszValue ) { _ASSERT( lpszEntry );
_ASSERT( lpszValue ); VERIFY( ERROR_SUCCESS == Editor_RegSetValue( m_hWnd, dwKey, pszConfig, lpszEntry, REG_SZ, (const
LPBYTE)lpszValue, (lstrlen( lpszValue ) + 1) * sizeof( TCHAR ), 0 ) ); } void WriteProfileString( LPCWSTR lpszEntry, LPCWSTR lpszValue )
{ TCHAR szFileName[MAX_PATH]; if( GetModuleFile( szFileName ) ) { WriteProfileString( EEREG_EMEDITORPLUGIN, szFileName,
lpszEntry, lpszValue ); } } void WriteProfileInt( DWORD dwKey, LPCWSTR pszConfig, LPCWSTR lpszEntry, int nValue ) { _ASSERT
( lpszEntry ); DWORD dwData = nValue; VERIFY( ERROR_SUCCESS == Editor_RegSetValue( m_hWnd, dwKey, pszConfig, lpszEntry,
REG_DWORD, (const LPBYTE)&dwData, sizeof( DWORD ), 0 ) ); } void WriteProfileInt( LPCWSTR lpszEntry, int nValue ) { TCHAR
szFileName[MAX_PATH]; if( GetModuleFile( szFileName ) ) { WriteProfileInt( EEREG_EMEDITORPLUGIN, szFileName, lpszEntry,
nValue ); } } void WriteProfileBinary( DWORD dwKey, LPCWSTR pszConfig, LPCWSTR lpszEntry, const LPBYTE lpBuf, DWORD
cbBufSize, bool bVariableSize ) { _ASSERT( lpszEntry ); _ASSERT( lpBuf ); VERIFY( ERROR_SUCCESS == Editor_RegSetValue( m_hWnd,
dwKey, pszConfig, lpszEntry, REG_BINARY, (const LPBYTE)lpBuf, cbBufSize, bVariableSize ? EE_REG_VARIABLE_SIZE : 0 ) ); } void
WriteProfileBinary( LPCWSTR lpszEntry, const LPBYTE lpBuf, DWORD cbBufSize, bool bVariableSize ) { TCHAR szFileName
[MAX_PATH]; if( GetModuleFile( szFileName ) ) { WriteProfileBinary( EEREG_EMEDITORPLUGIN, szFileName, lpszEntry, lpBuf,
cbBufSize, bVariableSize ); } } void EraseProfile() { TCHAR szFileName[MAX_PATH]; if( GetModuleFile( szFileName ) )
{ Editor_RegSetValue( m_hWnd, EEREG_EMEDITORPLUGIN, szFileName, NULL, REG_SZ, (const LPBYTE)NULL, 0, 0 ); } } void
EraseEntry( LPCWSTR lpszEntry ) { TCHAR szFileName[MAX_PATH]; if( GetModuleFile( szFileName ) ) { Editor_RegSetValue( m_hWnd,
EEREG_EMEDITORPLUGIN, szFileName, lpszEntry, REG_SZ, (const LPBYTE)NULL, 0, 0 ); } } }; HINSTANCE EEGetInstanceHandle()
{ _ASSERTE( _ETLData.m_hInstance != NULL ); return _ETLData.m_hInstance; } // buffer must be MAX_PATH character long. BOOL
GetModuleFile( LPTSTR szFileName ) { TCHAR szPath[MAX_PATH]; if( !GetModuleFileName( EEGetInstanceHandle(), szPath, _countof
( szPath ) ) ){ return FALSE; } TCHAR szBuf[MAX_PATH]; LPTSTR pszFile = szBuf; GetFullPathName( szPath, MAX_PATH, szBuf,
&pszFile ); LPTSTR p = _tcschr( pszFile, _T('.') ); if( p != NULL ) *p = _T('\0'); lstrcpyn( szFileName, pszFile, MAX_PATH ); return TRUE; }
WORD EEGetCmdID() { _ASSERTE( _ETLData.m_wCmdID >= EEID_PLUG_IN1 ); _ASSERTE( _ETLData.m_wCmdID <=
EEID_PLUG_IN1 + 255 ); return _ETLData.m_wCmdID; } CETLFrameX* GetFrameFromFrame( HWND hwndFrame ) { _ASSERTE
( hwndFrame ); _ASSERTE( IsWindow( hwndFrame ) ); ASSERT_STRICT( _ETLData.m_pETLFrameMap != NULL ); if
( _ETLData.m_pETLFrameMap == NULL ) return NULL; CETLFrameX* pFrame = NULL; { CETLLock lock; CETLFrameMap::iterator it
= _ETLData.m_pETLFrameMap->find( hwndFrame ); ASSERT_STRICT( it != _ETLData.m_pETLFrameMap->end() ); if( it !=
_ETLData.m_pETLFrameMap->end() ){ pFrame = it->second; _ASSERTE( pFrame != NULL ); } } return pFrame; } CETLFrameX*
GetFrame( HWND hwnd ) { _ASSERTE( IsWindow( hwnd ) ); HWND hwndFrame = GetAncestor( hwnd, GA_ROOTOWNER ); // for( ; ; ) { //
HWND hwndParent = GetParent( hwnd ); // if( hwndParent == NULL ){ // CETLFrameX* pFrame = GetFrameFromFrame( hwnd ); // return
pFrame; //// hwndFrame = hwnd; //// break; // } // hwnd = hwndParent; // } _ASSERTE( IsWindow( hwndFrame ) ); CETLFrameX* pFrame =
GetFrameFromFrame( hwndFrame ); // _ASSERTE( pFrame ); return pFrame; } CETLFrameX* GetFrameFromDlg( HWND hwnd ) { return
GetFrame( hwnd ); } CETLFrameX* GetFrameFromView( HWND hwndView ) { return GetFrame( hwndView ); } void DeleteAllFrames()
{ for( CETLFrameMap::iterator it = _ETLData.m_pETLFrameMap->begin(); it != _ETLData.m_pETLFrameMap->end(); it++ )
{ CETLFrameX* pFrame = it->second; pFrame->OnEvents( NULL, EVENT_CLOSE_FRAME, 0 ); delete pFrame; }
_ETLData.m_pETLFrameMap->clear(); } BOOL APIENTRY DllMain( HINSTANCE hModule, DWORD ul_reason_for_call,
LPVOID /*lpReserved*/ ) { if( ul_reason_for_call == DLL_PROCESS_ATTACH ){ #ifdef _DEBUG _CrtSetDbgFlag
( _CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF | _CRTDBG_CHECK_ALWAYS_DF ); #endif _ASSERTE
( _ETLData.m_hInstance == NULL ); _ETLData.m_hInstance = hModule; } else if( ul_reason_for_call == DLL_PROCESS_DETACH )
{ _ASSERTE( _ETLData.m_pETLFrameMap == NULL ); } return TRUE; } extern "C" void __stdcall OnCommand( HWND hwndView )
{ CETLFrameX* pFrame = GetFrameFromView( hwndView ); pFrame->OnCommand( hwndView ); } extern "C" BOOL __stdcall
QueryStatus( HWND hwndView, LPBOOL pbChecked ) { CETLFrameX* pFrame = GetFrameFromView( hwndView ); if( pFrame == NULL )
return FALSE; return pFrame->QueryStatus( hwndView, pbChecked ); } extern "C" UINT __stdcall GetMenuTextID() { return
CETLFrameX::GetMenuTextID(); } extern "C" UINT __stdcall GetStatusMessageID() { return CETLFrameX::GetStatusMessageID(); }
extern "C" UINT __stdcall GetBitmapID() { return CETLFrameX::GetBitmapID(); } extern "C" void __stdcall OnEvents( HWND hwndView,
UINT nEvent, LPARAM lParam ) { HWND hwndFrame = NULL; if( nEvent != EVENT_CLOSE ){ _ASSERTE( hwndView ); hwndFrame =
GetParent( hwndView ); _ASSERTE( hwndFrame ); } if( nEvent & EVENT_CREATE ){ _ASSERTE( (UINT)lParam >= EEID_PLUG_IN1
&& (UINT)lParam <= EEID_PLUG_IN1 + 255 ); _ASSERTE( _ETLData.m_wCmdID == 0 ); _ASSERTE( _ETLData.m_pETLFrameMap ==
NULL ); _ETLData.m_wCmdID = LOWORD( lParam ); _ETLData.m_pETLFrameMap = new CETLFrameMap; if( Editor_GetVersion
( hwndView ) < 5000 ){ // previous versions of EmEditor do not fire EVENT_CREATE_FRAME. OnEvents( hwndView,
EVENT_CREATE_FRAME, lParam ); } } else { ASSERT_STRICT( _ETLData.m_pETLFrameMap != NULL ); if
( _ETLData.m_pETLFrameMap != NULL ){ if( nEvent & EVENT_CREATE_FRAME ){ _ASSERTE( _ETLData.m_wCmdID == LOWORD
( lParam ) ); CETLFrameX* pFrame = static_cast(_ETLCreateFrame()); pFrame->m_hWnd = hwndFrame; { CETLLock lock; _ASSERTE
( _ETLData.m_pETLFrameMap->find( hwndFrame ) == _ETLData.m_pETLFrameMap->end() ); _ETLData.m_pETLFrameMap->insert
( std::pair( hwndFrame, pFrame ) ); pFrame->OnEvents( hwndView, nEvent, lParam ); } } else if( nEvent & EVENT_CLOSE_FRAME )
{ CETLFrameX* pFrame; { CETLLock lock; CETLFrameMap::iterator it = _ETLData.m_pETLFrameMap->find( hwndFrame ); _ASSERTE
( it != _ETLData.m_pETLFrameMap->end() ); pFrame = it->second; pFrame->OnEvents( hwndView, nEvent, lParam );
_ETLData.m_pETLFrameMap->erase( it ); } _ETLDeleteFrame( pFrame ); } else if( nEvent & EVENT_CLOSE ){ ASSERT_STRICT
( _ETLData.m_pETLFrameMap->empty() ); DeleteAllFrames(); // previous versions of EmEditor do not fire EVENT_CLOSE_FRAME. delete
_ETLData.m_pETLFrameMap; _ETLData.m_pETLFrameMap = NULL; _ETLData.m_wCmdID = 0; } else { CETLFrameX* pFrame;
{ CETLLock lock; CETLFrameMap::iterator it = _ETLData.m_pETLFrameMap->find( hwndFrame ); if( it ==
_ETLData.m_pETLFrameMap->end() ) { return; } pFrame = it->second; } pFrame->OnEvents( hwndView, nEvent, lParam ); } } } } extern
"C" LRESULT __stdcall PlugInProc( HWND hwnd, UINT nMsg, WPARAM wParam, LPARAM lParam ) { // hwnd can be either view handle,
frame handle, or plug-ins settings dialog handle. LRESULT lResult = 0; switch( nMsg ){ case EP_GET_BITMAP: lResult =
CETLFrameX::GetBitmap( wParam ); break; case EP_GET_MASK: lResult = CETLFrameX::GetMask( wParam ); break; case
EP_GET_NAMEA: lResult = CETLFrameX::GetNameA( (LPSTR)lParam, (size_t)wParam ); break; case EP_GET_NAMEW: lResult =
CETLFrameX::GetNameW( (LPWSTR)lParam, (size_t)wParam ); break; case EP_GET_VERSIONA: lResult = CETLFrameX::GetVersionA
( (LPSTR)lParam, (size_t)wParam ); break; case EP_GET_VERSIONW: lResult = CETLFrameX::GetVersionW( (LPWSTR)lParam, (size_t)
wParam ); break; case EP_GET_INFO: lResult = CETLFrameX::GetInfo( wParam ); break; default: { // hwnd is plug-ins settings dialog
handle or view window handle. // GetParent( hwnd ) is always Frame window handle. CETLFrameX* pFrame = GetFrame( GetParent
( hwnd ) ); if( pFrame ){ lResult = pFrame->PlugInProc( hwnd, nMsg, wParam, lParam ); } } break; } return lResult; }
```