# Software License Agreement

## HTML Add-on

version 26



3200 Maysilee Street

Austin, TX 78728

**512-733-2525**

## Software License Agreement

The Software is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The Software is licensed, not sold. This LICENSE AGREEMENT grants you the following rights:

A) This product is licensed per developer basis. Each developer working with this package needs to purchase a separate license.

B) When used this product within a desktop application, you are granted the right to modify and link the editor routine into your application. Such an application is free of distribution royalties with these conditions: the target application is not a stand-alone HTML viewer or a stand-alone HTML writer; the target application uses this product for one operating system platform only; and the source code (or part) of the product is not distributed in any form. As the HTML Add-on control needs TE Edit control to function, you also need to purchase a copy of TE Edit control. Sub Systems, Inc. reserves the right to prosecute anybody found to be making illegal copies of this software.

C. The DESKTOP LICENSE allows for the desktop application development. Your desktop application using this product can be distributed royalty-free. Each desktop license allows one developer to use this product on up to two development computers. A developer must purchase additional licenses to use the product on more than two development computers.

D. The SERVER LICENSE allows for the server application development. The server licenses must be purchased separately when using this product in a server application. Additionally, the product is licensed per developer basis. Only an UNLIMITED SERVER LICENSE allows for royalty-free distribution of your server applications using this product.

E. ENTERPRISE LICENSE: The large corporations with revenue more than $50 million and large government entities must purchase an Enterprise License. An Enterprise license is also applicable if any target customer of your product using the Software have revenue more than $500 million. Please contact us at info@subsystems.com for a quote for an Enterprise License.

F. Your license rights under this LICENSE AGREEMENT are non-exclusive. All rights not expressly granted herein are reserved by Licensor.

G. You may not sell, transfer or convey the software license to any third party without Licensor's prior express written consent.

H. The license remains valid for 12 months after the issue date. The subsequent year

license renewal cost is discounted by 20 percent from the license acquisition cost. The license includes standard technical support, patches and new releases.

I. You may not disable, deactivate or remove any license enforcement mechanism used by the software.

This software is designed keeping the safety and the reliability concerns as the main considerations. Every effort has been made to make the product reliable and error free. However, Sub Systems, Inc. makes no warranties against any damage, direct or indirect, resulting from the use of the software or the manual and can not be held responsible for the same. The product is provided 'as is' without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of suitability for a particular purpose. The buyer assumes the entire risk of any damage caused by this software. In no event shall Sub Systems, Inc. be liable for damage of any kind, loss of data, loss of profits, interruption of business or other financial losses arising directly or indirectly from the use of this product. Any liability of Sub Systems will be exclusively limited to refund of purchase price.

Sub Systems, Inc. offers a 30-day money back guarantee with the product. The money back guarantee is not available when the product is purchased with dll source.

# Disclaimer

This software is designed keeping the safety and the reliability concerns as the main considerations. Every effort has been made to make the product reliable and error free. However, Sub Systems, Inc. makes no warranties against any damage, direct or indirect, resulting from the use of the software or the manual and can not be held responsible for the same.

IBM PC,XT AND AT are the trademarks of International Business Machine.

MSDOS, Windows, Visual C++, MFC, and Visual Basic are the trademarks of Microsoft Corp. (for ease of reading Windows refer to MS Windows)

Delphi is the trademark of Borland International.

The Graphics Interchange Format(c) is the Copyright property of CompuServe Incorporated. GIF(sm) is a Service Mark property of CompuServe Incorporated.

# General Overview

This product allows your application to display an HTML document, navigate among documents or within the same document, edit HTML text, and convert RTF text to the HTML format. This product utilizes the TE Editor Control to actually display the text in a window. Therefore, the latest ter29.dll must be available to use this product.

**Picture Formats Support**: The product supports the following picture formats:

| | |
|---|---|
| BMP | MS Windows device independent bitmap format |
| WMF | MS Windows placeable metafile format |
| JPG | JPEG format |
| PNG | Portable Network Graphic format |
| TIFF | TIFF format |

In addition the product allows your application to translate other image formats which are not currently supported by HTML Add-on. Please refer to HTS_PICTURE message later in the manual.

Sub Systems also offers free support for GIF, GIF animation, and TIFF formats when you have a Unisys license to use these files.

# Getting Started

This chapter describes the contents of the software diskettes and provides a step by step process of incorporating the Hts DLL into your application.

**In This Chapter**



## Files

The package contains the DLL files, the source code files, the include files, resource files, and make files that are necessary to incorporate the Hts routine into your application. In addition, you will also need the latest version of ter29.dll (TE Edit control).

The package also includes a set of files to construct a demo program. The demo program shows by example the process of linking the DLL to your program.

**DLL Demo Files:**

The following demo files are included in the c_demo.zip file.

| | |
|---|---|
| DEMO.C | Source code for the demo program |

| | |
|---|---|
| DEMO.H | Include file for the demo program |
| DEMO.RC | Resource source file for the demo program |
| DEMO.DEF | Definition file for linking the demo program |
| DEMO.EXE | Executable demo program |
| DEMO_DLG.H | Dialog Identifiers for the demo program |
| DEMO_DLG.DLG | Dialog templates for the demo program |
| DEMO_DLG.RES | Compiled dialogs for the demo program |

**Hts Source Files:**

| | |
|---|---|
| Hts.C | Contain initialization and messaging functions. |
| Hts1.C | Contains editing functions. |
| HTS_IO.C | File input functions |
| HTS_IO1.C | File output functions |
| HTS_IMG.C | Image translation and decompression routines. |
| HTS_MISC.C | Other auxiliary functions.<br>(These source files with .C extension contain the Include statements for Hts.H, Hts1.H, WINDOWS.H and other runtime function headers) |
| Hts.H | The *include* file to include into your application module that calls the TER routine. It contains the constant definitions and the prototypes for the API functions. |
| Hts1.H | The *include* file used by the Hts modules. It contains global variables and some definition statements. All variable declaration statements are prefixed by 'PREFIX' word. The PREFIX word is defined in all source files except Hts.C as 'extern'. This strategy avoids the need for duplicate declarations for the main file and other auxiliary files that refer to the global variables as 'extern'. This file contains the Include statements for HTS_PROT.H and other header files. |
| HTS_PROT.H | Contains the prototypes for the internal Hts functions. |
| Hts.RC | Main resource file for the Hts routines. |

| | |
|---|---|
| Hts.RES | The compiled resource file required by the Windows' dialog editor. |
| Hts.DEF | The definition file to create the hts26.dll file. |
| hts26.dll | The DLL . |
| hts26.lib | Import library for hts26.dll |

**Visual Basic Interface and Demo Files:**

| | |
|---|---|
| Hts.BAS | Function declaration file. |
| DMO_VBX.FRM | Demo form file. |
| DMO_VBX.MAK | Demo make file. |

**Delphi Interface and Demo Files:**

| | |
|---|---|
| Hts.PAS | Global constant declaration file. |
| HTS_PROT.PAS | Function declaration file. |
| DMO_DLP.DPR | Demo project file. |
| DMO_DLP1.DFM | Demo form file. |
| DMO_DLP1.PAS | Demo code file. |

**Make Files:**

| | |
|---|---|
| MAKE-MC.BAT | Compiles and links the Hts routines using |
| MAKE-MC | Microsoft 'C'. |
| MAKE-BC.BAT | Compiles and links the Hts routines using |
| MAKE-BC | Borland 'C'. |

## License Key

*Your license key is e-mailed to you after your order for Html Add-on is processed.*

**When importing/exporting HTML text directly in TE Edit Control:**

When using Html Add-on only to import/export HTML text in TE Edit Control, please use the TerSetHtmlAddOnKey method exported by TE Edit Control at the beginning of your program:

TerSetHtmlAddOnKey("xxxxx-yyyyy-zzzzz")

Replace the 'xxxxx-yyyyy-zzzzz' by your Html Add-on license key. The license key for TE Edit Control will not work with the TerSetHtmlAddOnKey method.

Or, you can use the HtmlAddOnKey property of the Toc ActiveX control.

**When using HTML Add-on as an HTML Editor:**

You would set the license key for a stand-alone usage by using the HtsSetLicenseKey function. This should be preferably done before calling other methods.

HtsSetLicenseKey("xxxxx-yyyyy-zzzzz")

Replace the 'xxxxx-yyyyy-zzzzz' by your *Html Add-on* license key.

## Linking the Editor to Your Application

The package contains the hts26.lib file which must be linked to your application. You also need the ter29.lib to access the underlying editor functions. Modify the link statement of your application to include the hts26.lib and ter29.lib files into the list of libraries.

Your application modules which use Hts API functions, should also include the Hts.H file. The Hts.H file includes the TER.H file which is needed to interface with the ter29.dll

## Interfacing with the Hts DLL

The Hts and TER DLLs work together to create an HTML viewer/editor control. The Hts DLL performs the HTML related activities, whereas the TER DLL displays the text and processes the mouse messages. Your application needs to perform these steps to incorporate the Hts and TER DLLs into your application (please refer to the DEMO.C module for an example).

1. Create a window of the TER_CLASS. You can use any available method to create this window, such as using the CreateWindow SDK call, CreateTerWindow function (TER API), creating a TER control inside a dialog box, or creating a window using the MFC class wrappers such as CTer and CTerView classes. While creating a window using the CreateTerWindow function, you must specify an empty buffer because the Hts DLL needs an empty window to load a HTML document.

Word-wrap and Fitted View modes must always be turned on when creating a TER window. For a description of other viewing modes, please refer to the TE Edit control manual.

The ReadOnly flag determines the viewer or editor session. **Turn on the ReadOnly flag to create an HTML viewer.**

2. Call the HtsInitialize API to initialize the HTML operational variables for the newly created window. Example:

   HtsInitialize(hTerWin, hParentWin);

   The specified parent window receives the HTML messages.

3. Read an HTML document into the window. Example:

   HtsRead(hTerWin, HTML_FILE, filename, NULL, 0, NULL);

   The above statement reads an HTML disk file into the window.

   Please refer to the *Application Interface Function* chapter for a detailed description of these API functions.

   The parent window of your application also needs to handle these messages:

   | | |
   |---|---|
   | HTS_TITLE: | This message contains the document title. Your application will typically display this title as the window caption. |
   | HTS_LINK: | This message prompts your application to load another HTML document. |
   | HTS_PICTURE: | This message is sent to resolve the path for a picture element currently being loaded into the window. |
   | HTS_FORM: | This message is fired when the user submits a form. |

   Please refer to the *Message Communication* chapter for a detailed description of these messages.

4. If you are invoking an HTML editing session, use the HtsCommand function within your application menu to invoke various editing functions.

# Application Interface functions

These API functions allow you to load and manipulate HTML data in an existing TER window. Your application must include the Hts.H file, which defines these functions.

Most functions in this chapter need the first argument as the window handle (hWnd) of the control. This parameter requires further description. When the control contains a normal HTML document without the embedded frames, the hWnd parameter is used by

the function without modification.

However, when the control contains a document containing embedded frames, the hWnd parameter is modified internally. Unless specified, your program still needs to supply the window handle of the control as the 'hWnd' parameter. The 'hWnd' parameter that your program supplies can be modified as following:

> If the user clicked on a hyperlink text, the editor remembers the target frame (a part of the href specification) name for the hyperlink. The editor uses the window handle of the target frame instead of the one supplied by your program. To reset the target frame, please use the HtsSetTarget function. The target frame is also reset automatically when your program reads the next (link) document using the HtsRead function. This feature allows you to automatically load the link document into the correct frame window.
> When the user clicks inside a frame window, that frame window is treated as the active window. The editor uses the window handle of the active window instead of the one supplied by your program. This feature ensures that the API functions are applied to the content of the current active window.

The following is a description of the Hts API functions in an alphabetic order:

*Note: The API provided in this help file is now superseded by the functions in TE. Please use the TE functions instead.*

**In This Chapter**

## HtsAddSelectionItem

**Add an item to the selection box.**

BOOL HtsAddSelectionItem(hWnd, CtlId, text, value, selected, insert)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int CtlId; | The control id of the current selection field. Set this parameter to -1 to access the selection box at the current cursor location. |
| LPBYTE text; | Text string to display for the item. Set this field to NULL to prompt the user for parameters. |
| LPBYTE value; | Value string for the item. This parameter can be set to NULL. |
| BOOL selected; | Set to TRUE to select the new item after it is inserted. |
| BOOL insert; | Set to TRUE to insert the new item before the current item. Set to FALSE to append the new item after the last item. |

**Return Value:** This function returns TRUE if successful.

**See Also:**
HtsInsertSelectField
HtsDelSelectionItem

## HtsClearWindow

**Clear the current window:**

BOOL HtsClearWindow(hWnd)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |

**Description:** Use this function to remove existing text from the given window. This function also releases resources used by the previous text. This function is usually called before reading a new HTML document into a window

**Return Value:** This function returns TRUE if successful.

**See Also:**
HtsRead

## HtsClose

**Close the current window after saving modified data.**

BOOL HtsClose(hWnd, OutputTo, FileName, hBuf, BufLen)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |

| | | |
|---|---|---|
| int OutputTo; | Output to: | |
| | HTML_FILE: | Specify the output file name using the FileName argument. |
| | HTML_BUF: | The output buffer is returned via the hBuf and BufferLen parameters. |
| LPBYTE FileName; | Output file name. The file name may be prefixed by a subdirectory path. Set to NULL when the 'OutputTo' argument is HTML_BUF. | |
| HGLOBAL far *hBuf; | When saving to a buffer, this pointer receives the global memory handle of the buffer containing the text. | |
| LPLONG BufLen; | When saving to a buffer, this pointer receives the buffer length. | |

**Description:** This function calls the HtsSave function to save the text if modified. Then it calls the CloseTer function to close the TER window. This function is meant to be called from your application's 'close' or 'quit' menu options.

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsSave

## HtsCommand

**Execute a menu command.**

BOOL HtsCommand(hWnd, MenuId)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int MenuId; | The menu id for the command to execute. |

**Description:** The following predefined menu ids are available:

| | |
|---|---|
| HTM_ADDRESS | Set the 'Address' attribute for the paragraph. |
| HTM_ADD_SELECT_ITEM | Add items to the current selection (list/combo) box. |
| HTM_BLOCKQUOTE | Set the 'Blockquote' attribute for the paragraph. |
| HTM_BOLD | Set bold character style. |
| HTM_CENTER | Center the paragraph. |

| | |
|---|---|
| HTM_CHAR_NORMAL | Reset all character styles. |
| HTM_CHAR_TYPE_NORMAL | Reset miscellaneous character types such as Example, Variable, Sample, etc. |
| HTM_COPY | Copy the selected text to clipboard. |
| HTM_CUT | Cut the selected text to clipboard. |
| HTM_DEL_SELECT_ITEM | Delete the selected items from the current selection (list/combo) box. |
| HTM_EDIT_FORM_ID | Edit the parameters for a form id. |
| HTM_EDIT_LINK | Edit the link parameters. |
| HTM_EDIT_RULE | Edit parameters for the current horizontal rule. |
| HTM_EDITOR | Switch to the editor mode. |
| HTM_EXAMPLE | Set the 'Example' type to text. |
| HTM_FONT_COLOR | Set font color |
| HTM_FONT_SIZE | Set font size |
| HTM_FONTS | Edit the font attribute for the viewer. |
| HTM_HDR1 | Apply the 'Header 1' attribute to the paragraph. |
| HTM_HDR2 | Apply the 'Header 2' attribute to the paragraph. |
| HTM_HDR3 | Apply the 'Header 3' attribute to the paragraph. |
| HTM_HDR4 | Apply the 'Header 4' attribute to the paragraph. |
| HTM_HDR5 | Apply the 'Header 5' attribute to the paragraph. |
| HTM_HDR6 | Apply the 'Header 6' attribute to the paragraph. |
| HTM_HDR7 | Apply the 'Header 7' attribute to the paragraph. |
| HTM_INSERT_BUTTON_FIELD | Insert a button or checkbox field. |
| HTM_INSERT_IMAGE_FIELD | Insert an image field. |
| HTM_INSERT_LINK | Insert a hyperlink. |

| | |
|---|---|
| HTM_INSERT_PICT | Insert a picture. |
| HTM_INSERT_RULE | Insert a horizontal rule. |
| HTM_INSERT_SELECT_FIELD | Insert a selection (list/combo) box. |
| HTM_INSERT_TEXT_FIELD | Insert a text line, text area, or password field. |
| HTM_ITALIC | Set the italic style. |
| HTM_JUSTIFY | Justify the paragraph |
| HTM_KBD | Set the Keyboard character type. |
| HTM_LEFT | Left justify the paragraph. |
| HTM_LINK | Edit the color and style of link text. |
| HTM_LIST_DD | Set the 'Definition Data' attribute for the paragraph. |
| HTM_LIST_DT | Set the 'Definition Term' attribute for the paragraph. |
| HTM_LIST_LVL1 | Set level 1 for the list item. |
| HTM_LIST_LVL2 | Set level 2 for the list item. |
| HTM_LIST_LVL3 | Set level 3 for the list item. |
| HTM_LIST_LVL4 | Set level 4 for the list item. |
| HTM_LIST_LVL5 | Set level 5 for the list item. |
| HTM_LIST_LVL6 | Set level 6 for the list item. |
| HTM_LIST_LVL7 | Set level 7 for the list item. |
| HTM_LIST_OL | Set the 'Ordered List' attribute for the paragraph. Please note that numbering for the ordered list is not available in the 'Edit' mode. |
| HTM_LIST_UL | Set the ' Unordered List' attribute for the paragraph. |
| HTM_NEW | Save any modifications and clear the window. |
| HTM_NEW_LIST_ITEM | Toggle the 'New Item' property of a list item. |
| HTM_OPEN | Save any modifications and open a new file. |

| | |
|---|---|
| HTM_PARA_FORM | Mark a text area as a form. |
| HTM_PARA_NORMAL | Reset paragraph attributes such as justification, header, list, miscellaneous types, etc. |
| HTM_PARA_SPACE | Toggle space before and after the paragraph. |
| HTM_PRE | Set the 'preformatted' attribute for the paragraph. |
| HTM_PRINT | Print the text. |
| HTM_PRINT_PREVIEW | Print preview. |
| HTM_REFORMAT | Reformat the document. |
| HTM_RIGHT | Right justify the paragraph. |
| HTM_SAMP | Set the 'Sample' attribute for the text. |
| HTM_SAVE | Save the current file. |
| HTM_SAVE_AS | Save the current file in another name. |
| HTM_SET_FORM_ID | Set the form id for the current form. |
| HTM_SHOW_HIDDEN | Show the hidden text. |
| HTM_SHOW_PARA_MARK | Show the paragraph markers. |
| HTM_STRIKE | Set the 'strike' style for the text. |
| HTM_SUBSCR | Set the 'subscript' style for the text. |
| HTM_SUPSCR | Set the 'superscript' style for the text. |
| HTM_TBL_APPEND_COL | Append a table column. |
| HTM_TBL_APPEND_ROW | Append a table row. |
| HTM_TBL_CREATE | Create a table. |
| HTM_TBL_DELETE_COL | Delete the current table column or all selected table columns. |
| HTM_TBL_DELETE_ROW | Delete the current table row or all selected table rows. |
| HTM_TBL_INSERT_COL | Insert a table column before the current column. |

| | |
|---|---|
| HTM_TBL_INSERT_ROW | Insert a table row before the current row. |
| HTM_TT | Set the 'Typewriter' character type. |
| HTM_ULINE | Underline the text. |
| HTM_VAR | Set the 'variable' character type. |
| HTM_VIEWER | Switch to the viewer mode. |

**Return Value:** This function returns FALSE if an incorrect menu id is specified, otherwise it returns TRUE.

**See Also:**
HtsMenuSelect

## HtsDelSelectionItem

**Delete the selected items from the selection box.**

BOOL HtsDelSelectionItem(hWnd, CtlId)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int CtlId; | The control id of the current selection field. Set this parameter to -1 to access the selection box at the current cursor location. |

**Description:** If more than one item is selected in the selection box, all selected items will be deleted.

**Return Value**: This function returns TRUE if successful.

**See Also:**
HtsInsertSelectField
HtsAddSelectionItem

## HtsEditFormId

**Edit or create a form id.**

int HtsEditFormId(hWnd, new, id, method, action, ShowDialog)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |

| | |
|---|---|
| BOOL new; | Set to TRUE to create a new form id. |
| int id; | When the 'new' parameter is FALSE, the 'id' parameter specifies the form id to edit. |
| int method; | Method: METHOD_GET or METHOD_POST |
| LPBYTE action; | Action string for the form id. |
| BOOL ShowDialog; | Set to TRUE to prompt the user for the parameters. |

**Description:** A form id is associated with every form. You will typically create a form id before creating a form.

**Return Value:** The function returns the form id when successful, otherwise it returns -1.

**See Also:**
HtsSetFormId
HtsSetForm

## HtsEditRule

**Edit the horizontal rule parameters.**

BOOL HtsInsertRule (hWnd, width, thickness, align, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int width; | Width of the rule in terms of percentage of the screen width. |
| int thickness; | Thickness of the rule in points. |
| int align; | Rule alignment: LEFT, CENTER, RIGHT_JUSTIFY. |
| BOOL repaint; | Repaint the screen after this operation. |

**Description:** This functions edits the parameters for the horizontal rule at the current cursor location. Set the width or the thickness parameter to 0 to invoke a user dialog box to accept the parameters.

**Return Value:** The function returns a TRUE value when successful.

**See Also:**
HtsInsertRule

## HtsGetActiveWnd2

**Retrieve the window handle of the active frame.**

HWND HtsGetActiveWnd(hWnd)

HWND HtsGetActiveWnd2(hWnd,FrameName)

| | |
|---|---|
| HWND hWnd; | The window handle of the control |
| LPBYTE FrameName; | The frame name to get the wnidow handle |

**Description:** This function is useful for determining the current active frame (or the frame with the specified name) within a document that contains embedded frames.

**Return Value:** This function returns the window handle of the frame that currently has the focus.

**See Also:**
HtsSetActiveWnd

## HtsGetCurForm

**Retrieve the form id at the current cursor position.**

int HtsGetCurForm(hWnd, TotalForms)

| | |
|---|---|
| HWND hWnd; | The window handle of the control |
| LPINT TotalForms; | The pointer to receive the total number of forms in the document. |

**Description:** This function is useful for determining the current form id and the total number of forms in the document.

**Return Value:** This function returns the form id of the current form. It returns -1 if the current line does not belong to a form. It also returns -1 if the current form does not contain any fields.

**See Also:**
HtsGetFormInfo

## HtsGetFont

**Get font information about an HTML style element.**

BOOL HtsGetFont(hWnd, font, typeface, PointSize, style, TextColor, TextBkColor)

| | |
|---|---|
| HWND hWnd; | The window handle of the control |
| int font; | Font id for an HTML style element. Refer to the description for a list of the font ids. |
| LPBYTE typeface; | The pointer to receive the font typeface. This buffer should be able to hold up to 32 characters. |
| LPINT PointSize; | The pointer to receive the point size of the font. |
| UINT far *style; | The pointer to receive the style bits for the font: |

| | |
|---|---|
| BOLD: | Bold |
| ITALIC: | Italic |
| ULINE: | Underline |
| STRIKE | Strikethrough |

| | |
|---|---|
| COLORREF far *TextColor; | The pointer to receive the text foreground color. |
| COLORREF far *TextBkColor; | The pointer to receive the text background color. |

**Description:** This function can be used to get font and color information about an HTML style element. The desired style element is specified using the 'font' argument. The following style elements can be specified:

| | |
|---|---|
| FONT_TEXT: | Body Text |
| FONT_H1: | Header 1 |
| FONT_H2: | Header 2 |
| FONT_H3: | Header 3 |
| FONT_H4: | Header 4 |
| FONT_H5: | Header 5 |
| FONT_H6: | Header 6 |
| FONT_H7: | Header 7 |
| FONT_ADDRESS: | Address |
| FONT_QUOTE: | Block Quote |

| | |
|---|---|
| FONT_MENU: | Menu List |
| FONT_DIR: | Directory List |
| FONT_PRE: | Preformatted Text |
| FONT_EXAMPLE: | Example |
| FONT_VAR: | Variables |
| FONT_KBD: | Keyboard Input |
| FONT_SAMP: | Sample Text |
| FONT_HV: | Helvetica Font |
| FONT_TR: | Times Roman |

**Return Value:** This function returns a TRUE value when successful.

**Example:**

```
BYTE typeface[32];
int PointSize;
UINT style;
COLORREF TextColor, TextBkColor;
HtsGetFont(hWnd, FONT_H1, typeface, &PointSize, &style,
           &TextColor, &TextBkColor);
// The above example retrieves the current font information
   for the 'Header1' style element.
```

**See Also:**
HtsSetFont
HtsGetLinkInfo

## HtsGetFontSize

**Get font size at the current cursor location.**

int HtsGetFontSize(hWnd, IsRelative)

| | |
|---|---|
| HWND hWnd; | The window handle of the control |
| LPINT IsRelative; | (output) This location receives a TRUE value (1) if the font size is relative, or a FALSE value (0) if the font size is absolute. |

**Return Value:** This function returns the HTML font size (1 to 7) at the current cursor location. A value of 0 indicates an error.

**See Also:**
HtsSetFontSize

## HtsGetFieldData

**Get the data for a field in a form.**

int HtsGetFieldData(hWnd, FormId, FieldId, name, IntData, TextData, TextDataLen)

| | |
|---|---|
| HWND hWnd; | The window handle of the control |
| int FormId; | The form id which contains the field. Set the FormId and FieldId parameters to –1 to get the information about the last 'submit' field clicked by the user. |
| int FieldId; | The field number for which to retrieve data. |
| LPBYTE name; | (OUTPUT) The name of the field. |
| LPINT IntData; | (OUTPUT) The pointer to receive the integer data. |
| LPBYTE TextData; | (OUTPUT) The pointer to receive the text data. |
| int TextDataLen; | The size of the TextData buffer. |

**Description:** This function retrieves the information about the specified field id in the specified form. This function is called by your application when it receives the HTS_FORM message.

The 'name' argument receives the field name. The IntData and the TextData pointers receive additional information about the field. This information varies for each field type. The following describes the usage for the IntData and TextData parameters by field type:

*FTYPE_TEXT, FTYPE_PASSWORD, or FTYPE_TEXTAREA:*

The TextData parameter receives the text data for the control. The IntData parameter is not used.

*FTYPE_RADIO or FTYPE_CHECKBOX*

The IntData is non-zero if the control is checked, otherwise it is zero. The TextData parameter is not used.

*FTYPE_HIDDEN:*

The TextData parameter receives the hidden text value. The 'IntData' parameter is not used.

*FTYPE_IMAGE:*

The TextData parameter receives the x and y pixel positions in the X,Y format. Example, a x=20 and y=10 pixel position will return a string containing "20,10". The IntData parameter is not used.

*FTYPE_SELECT:*

The TextData parameter receives the selections. If more than one item is selected, each item is separated by a '|' delimiter. The IntData parameter is not used.

**Return Value:** When successful, this function returns the field type:

| | |
|---|---|
| FTYPE_TEXT: | Single line text control. |
| FTYPE_RADIO: | Radio button. |
| FTYPE_SUBMIT: | 'Submit' push button. |
| FTYPE_RESET: | 'Reset' push button. |
| FTYPE_TEXTAREA: | Multiline text control. |
| FTYPE_CHECKBOX: | Check box. |
| FTYPE_HIDDEN: | Hidden text. |
| FTYPE_PASSWORD: | Single line password text control. |
| FTYPE_IMAGE: | Image. |
| FTYPE_SELECT: | Combo box or a list box. |

This function returns -1 when an error occurs.

**Example:**

```
BYTE action[100],method[10],name[100],TextData[300];
int i,TotalFields,FieldType,IntData;
TotalFields=HtsGetFormInfo(hWnd, 0, action,method);
  // Retrieve information about a form id 0.
for (i=0;i<TotalFields;i++) {
  FieldType=HtsGetFieldData(hWnd,0,i,name,&IntData,
                            TextData,299);
}HTS_FORM (message)
```

**See Also:**
HtsGetFormInfo

## HtsGetFormInfo

**Get the information about a form.**

int HtsGetFormInfo(hWnd, FormId, action, method)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int FormId; | The form id for which to retrieve the information. |
| LPBYTE action; | (output) This fields receives the 'action' text for the form. |
| LPBYTE method; | (output) This field receives the 'method' text for the form. The method text string is either 'GET' or 'POST'. |

**Description:** This function is typically called when your application receives the HTS_FORM message.

**Return Value:** On successful return, this function returns the total number of fields in the form. Otherwise it returns -1.

**Example:**

```
BYTE action[100],method[10];
int TotalFields;
TotalFields=HtsGetFormInfo(hWnd, 0, action,method);
  // Retrieve information about a form id 0.
```

**See Also:** HTS_FORM (message)

**See Also:**
HtsGetFieldData

## HtsGetFrameName

**Retrieve the current frame name.**

BOOL HtsGetFrameName(hWnd, name)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed. This window handle is used as given without any modification described in the beginning of this chapter. |
| LPBYTE name; | The location to receive the current frame name. |

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsSetFrameName

## HtsGetLinkDataEx

**Retrieve the parameters for the link at the cursor position.**

BOOL HtsGetLinkData(hWnd, href, name)

BOOL HtsGetLinkDataEx(hWnd, href, name, target)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| LPBYTE href; | The pointer to receive the 'href' parameter. |
| LPBYTE name; | The pointer to receive the 'name' parameter. |
| LPBYTE target; | The pointer to receive the frame target for the link. |

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsUpdateLinkEx

## HtsGetLinkInfo

**Get the color and style attributes for the link text.**

BOOL HtsGetLinkInfo(hWnd, style, TextColor)

| | | |
|---|---|---|
| HWND hWnd; | The handle of the window to be accessed | |
| UINT far *style; | The pointer to receive the style bits for the link font: | |
| | BOLD: | Bold |
| | ITALIC: | Italic |
| | ULINE: | Underline |
| | STRIKE | Strikethrough |
| COLORREF far *TextColor; | The pointer to receive the link foreground color. | |

**Description:** This function is used to retrieve the text color and style information for the link text.

**Return Value:** This function returns a TRUE value when successful.

**Example:**

```
UINT style;
COLORREF TextColor;
HtsGetLinkInfo(hWnd, &style, &TextColor);
```

**See Also:**
HtsSetLinkInfo
HtsGetFont

## HtsGetParaAttrib

**Get current paragraph attributes.**

BOOL HtsGetParaAttrib(hWnd, HdrLevel, ListType, ListLevel, ParaAuxId, MiscType)

| | | |
|---|---|---|
| HWND hWnd; | The handle of the window to be accessed | |
| LPINT HdrLevel; | This pointer returns the header level. A header level of 0 indicates a non-header paragraph. | |
| LPINT ListType; | This pointer returns the list type: | |
| | PARA_LIST_NONE: | Not a list. |
| | PARA_LIST_OL: | Ordered list. |
| | PARA_LIST_UL: | unordered list. |
| | PARA_LIST_DIR: | Directory list. |
| | PARA_LIST_MENU: | Menu list. |
| | PARA_LIST_DL: | Definition list. |
| LPINT ListLevel; | This pointer returns the list level for the list type of paragraph. A list level of 0 indicates a non-list type of paragraph. | |
| UINT far *ParaAuxId; | Addition paragraph attribute bits: | |
| | PARA_FORM: | This is paragraph belongs to a form. |
| | PARA_BEGIN_LI: | First paragraph of a list item. |
| | PARA_DT: | When ListType is set to PARA_LIST_DL, then this bit |

| | | |
|---|---|---|
| | | indicates the definition term for the definition list. |
| | Use the logical OR operator to check the attribute bits. | |
| LPINT MiscType: | Miscellaneous types: | |
| | PARA_BLOCKQUOTE: | Block quote. |
| | PARA_ADDRESS: | Address. |
| | PARA_HR: | Horizontal rule. |
| | PARA_PRE: | Preformatted text. |

**Return Value:** This function returns TRUE when successful.

## HtsGetTagData

**Retrieve the tag data.**

int HtsGetTagData(hWnd, type, id, start, int1, int2, str1, str2)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int type; | Tag type |
| int id; | Tag id |
| int start; | Index to begin the search from. Set to 0 to search the entire tag table. |
| LPINT int1; | The location to return the first integer data associated with this tag. |
| LPINT int2; | The location to return the second integer data associated with this tag. |
| LPBYTE str1; | The location to return the first string data associated with this tag. |
| LPBYTE str2; | The location to return the second string data associated with this tag. |

**Description:** The HTML add-on stores certain object data into a tag table. This function allows you to search the tag table and retrieve the associated tag values. A tag can have

up to 2 integer values (int1 and int2) and up to 2 string values (str1 and str2). Not all tags use all the values.

This table displays the available tag types, the associated id value and the data values:

| | | | | Values | |
|---|---|---|---|---|---|
| **Tag Type** | **Tag Id** | **Int1** | **Int2** | **String1** | **String2** |
| TAG_CELL_QFN | Cell Id | | | Qfn String | |
| TAG_META | HMETA_HTTP_EQUIV | | | http-equiv | content |
| TAG_META | HMETA_NAME | | | name | content |
| TAG_USEMAP | Picture id | | | 'Usemap' string | |

**Return Value:** The function returns an index into the tag table if the specified tag type/id is found. It returns -1 to indicate that the specified tag type/id was not found in the tag table.

**Example:**

```
int  int1, int2,
BYTE  string1[300],string2[300];
HtsGetTagData(hWnd,TAG_USEMAP,PictId,0,int1,int2,
  string1,string2);
```

This function returns the 'usemap' string for the specified picture id in the 'string1' variable.

## HtsGetTarget

**Retrieve the current target frame for the hyperlink jump.**

BOOL HtsGetTarget(hWnd, target)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| LPBYTE target; | The pointer to receive the current target frame name. |

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsSetTarget

## HtsGetTopWin

**Get the window handle of the top window in the current frame set.**

HWND HtsGetTopWin(hWnd)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |

**Description:** A framed html document is displayed using one or more child (frame) windows which are contained within the top window. This function is useful in determining the window handle of the top window.

**Return Value:** This function returns the window handle of the top window in the current frame set. It returns a NULL to indicate an error condition.

## HtsGetUserTag

**Get the text data for the user tag.**

BOOL HtsGetUserTag(hWnd, TagId, TagText)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int TagId; | Tag id for the user tag to retrieve |
| LPBYTE TagText; | The string to receive the tag data. |

**Description:** This function retrieves the text data associated with the given user tag id.

**Return Value:** This function returns TRUE if the user tag is found. Otherwise it returns FALSE.

**See Also:**
HtsSetUserTag

## HtsInitialize

**Initialize the Hts operating variables.**

BOOL HtsInitialize(hWnd, hParentWnd)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| HWND hParentWnd; | The parent of the TER window to receive the messages. |

Your application needs to receive messages from the Hts DLL. Therefore, this argument must specify a valid parent window handle.

**Description:** This function must be called immediately after creating a TER window and before calling any other Hts API function. This routine performs the following functions:

Request space allocation from the TER window to hold the Hts DLL variables.
Initialize the Hts DLL variables.
Initialize and create fonts for the HTML style elements.
Modify the TER environment variables.
Set a callback from the TER DLL to receive the mouse messages.

**The HtsInitialize function should not be used when using TE Edit Control functions such as GetTerBuffer, SetTerBuffer, ReadTerFile, SaveTerFile, and the Data property to import or export HTML text.**

**Return Value:** The function returns a TRUE value when successful.

**Example:**

```
HWND hWnd=CreateWindows(.........);    // create a window
                                          of TER_CLASS

HtsInitialize(hWnd,hParentWnd);
```

## HtsInsertButtonField

**Insert a button or check-box field into form.**

BOOL HtsInsertButtonField(hWnd, name, type, checked, value, repaint)

| | | |
|---|---|---|
| HWND hWnd; | The handle of the window to be accessed | |
| LPBYTE name; | The name of the text field (null terminated string). Set this parameter to NULL to prompt the user for the parameters. | |
| int type; | The field type can be one of the following values: | |
| | FTYPE_RADIO: | Radio button |
| | FTYPE_CHECKBOX: | Check-box field |
| | FTYPE_SUBMIT: | Submit button |
| | FTYPE_RESET: | Reset button |
| BOOL checked; | For the radio button and check-box fields, this parameter specifies the initial 'check' status of the field. | |

LPBYTE value; Field value string. This parameter can be set to NULL.

BOOL repaint; Repaint the screen after this operation.

**Return Value:** If successful, the function returns the control id of the new field. Otherwise it returns 0. You can use the Windows SDK function GetDlgItem to translate the control id into the window handle of the control.

**See Also:**
HtsInsertTextField
HtsInsertSelectField

## HtsInsertLink

**Insert a hyperlink at the current location.**

BOOL HtsInsertLink(hWnd, LinkText, LinkPict, LinkHref, LinkName, IsMap, repaint)

HWND hWnd; The handle of the window to be accessed

LPBYTE LinkText; Link text (null terminated) string for a text type link.

LPBYTE LinkPict; picture file name for a picture type link. This function supports BMP, WMF, PNG, and JPEG picture file formats.

*In addition, Sub Systems also offers free additional support for GIF and TIFF picture files when you have a Unisys license to use these files.*

*Only one of the above parameters (LinkText or LinkPict) may be used.*

*If both the 'LinkText' parameter and the 'LinkPict' parameters are NULL, then this function will display a dialog box for the user to enter the parameters.*

LPBYTE LinkHref; Href string (NULL terminated) for the link. This parameter should contain the url for the link

LPBYTE LinkName; Link name (NULL terminated) string.

*At least one of the 'LinkHref' or 'LinkName' parameters must be provided.*

BOOL IsMap; Set to TRUE to transmit mouse click location on the picture (on a picture link). This parameter is not used for a text link. (default is FALSE).

BOOL repaint; Repaint the screen after this operation.

**Return Value:** The function returns a TRUE value when successful.

**See Also:**
HtsInsertPicture

## HtsInsertPicture

**Insert a picture at the current location.**

int HtsInsertPicture(hWnd, file, align, IsMap, IsInput, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| LPBYTE file; | picture file name. This function supports BMP, WMF, PNG, and JPEG picture file formats. You can set this parameter to NULL to invoke a dialog box to accept user parameters.<br>In addition, Sub Systems offers free support for GIF and TIFF formats when you have a Unisys license to used these formats. |
| int align; | Picture alignment: ALIGN_TOP, ALIGN_BOT, ALIGN_MIDDLE, HALIGN_LEFT, HALIGN_RIGHT. Use 0 for default. |
| BOOL IsMap; | Set to TRUE to transmit mouse click location on the picture. (default is FALSE). |
| BOOL InInput; | Set to TRUE to create a 'Send' input field within the current form. (default is FALSE). |
| BOOL repaint; | Repaint the screen after this operation. |

**Return Value:** The function returns a non-zero picture id when successful. Otherwise it return zero.

**See Also:**
HtsInsertLink

## HtsInsertRule

**Insert a horizontal rule.**

BOOL HtsInsertRule (hWnd, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |

BOOL repaint; Repaint the screen after this operation.

**Return Value:** The function returns a TRUE value when successful.

**See Also:**
HtsEditRule

## HtsInsertSelectField

**Insert a selection box into form.**

BOOL HtsInsertSelectField(hWnd, name, MultiSelect, NumLines, repaint)

HWND hWnd; The handle of the window to be accessed

LPBYTE name; The name of the text field (null terminated string). Set this parameter to NULL to prompt the user for the parameters.

BOOL MultiSelect; Set to TRUE to enable the selection of multiple items inside the list box.

int NumLines; Number of visible lines in the selection box. To create a combo-box, set this field to 1. To create a list box, set this parameter to a value greater than one.

BOOL repaint; Repaint the screen after this operation.

**Description:** This function creates an empty selection box. You can use the HtsAddSelectionItem function to add items in the selection box.

**Return Value:** If successful, the function returns the control id of the new field. Otherwise it returns 0. You can use the Windows SDK function GetDlgItem to translate the control id into the window handle of the control.

**See Also:**
HtsAddSelectionItem
HtsDelSelectionItem
HtsInsertTextField

## HtsInsertTextField

**Insert a text field into form.**

BOOL HtsInsertTextField(hWnd, name, type, NumCols, MaxCols, NumRows, InitText, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| LPBYTE name; | The name of the text field (null terminated string). Set this parameter to NULL to prompt the user for the parameters. |
| int type; | The field type can be one of the following values: |

| | |
|---|---|
| FTYPE_TEXT: | Text line field |
| FTYPE_TEXTAREA: | Text area field |
| FTYPE_PASSWORD: | Password field. |

| | |
|---|---|
| int NumCols; | The width of the field in terms of number of characters. |
| int MaxCols; | Maximum number of characters allowed in the field. |
| int NumRows; | For a text area field (FTYPE_TEXTAREA), this parameter indicates the number of lines in the text box. |
| LPBYTE InitText; | Initialize text string. This parameter can be set to NULL. |
| BOOL repaint; | Repaint the screen after this operation. |

**Return Value:** If successful, the function returns the control id of the new field. Otherwise it returns 0. You can use the Windows SDK function GetDlgItem to translate the control id into the window handle of the control.

**See Also:**
HtsInsertButtonField

## HtsInternetGet

**Download a file from internet.**

BOOL HtsInternetGet(hWnd, url, OutFile, GetFromCache)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| LPBYTE url; | The url of the internet file |
| LPBYTE OutFile; | The name of the local file to save the internet file as. |
| BOOL GetFromCache; | Set to FALSE to always download the file. Set to TRUE to copy the file from cache, if available. |

**Return Value:** This function returns TRUE when successful. Otherwise, it returns a FALSE value.

## HtsInViewMode

**Check if the current document is being displayed in the View Mode.**

BOOL HtsInViewMode(hWnd)

HWND hWnd; The handle of the window to be accessed

**Return Value:** This function returns TRUE if the current document is being displayed in the View Mode.

## HtsIsHttpFile

**Check if the given file is an internet file.**

BOOL HtsIsHttpFIle(FileName)

LPBYTE FIleName; The file path to check

**Return Value:** This function returns TRUE if the given file path resides on the internet.

**See Also:**
HtsIsLoaclFile
HtsSetDownloadDir
HtsInternetGet

## HtsIsLoaclFile

**Check if the given file is a local file.**

BOOL HtsIsLocalFIle(FileName)

LPBYTE FIleName; The file path to check

**Return Value:** This function returns TRUE if the given file path resides on a local disk.

**See Also:**

HtsIsHttpFile
HtsSetDownloadDir
HtsInternetGet

## HtsLoadControl

**Load the Hts DLL:**

BOOL HtsLoadControl()

**Description:** This function is used to load the Hts DLL. This function ensures that the Hts DLL is actually linked with your application.

**Return Value:** This function always returns a TRUE value.

## HtsLparam2String

**Copy text from a long parameter (lParam) to a string variable**

BOOL HtsLparam2String(lParam, string)

| | |
|---|---|
| long lParam; | Source pointer in the long form. |
| LPBYTE string; | Destination pointer |

**Description:** This function is used by a Visual Basic application to copy the text from a long parameter to a string variable.

Return Value: This function always returns a TRUE value.

**Example:**

```
sub html(message as Integer, wParam as Integer,
                             lParam as Long)
  Dim text as string
  text=space$(300)    ' allocate 100 bytes for
                        the text variable
  HtslParam2String(lParam,text)  ' copy the text
                                   from lParam to text
  text="new text"
  HtsString2Lparam(text,lParam) ' return the new string
end sub
```

**See Also:**
HtsString2Lparam

## HtsMenuEnable

**Return the enable/disable status of a menu item.**

BOOL HtsMenuEnable(hWnd, MenuId)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int MenuId; | The menu id to return the status. Please refer to the 'HtsCommand' function for a list of available menu ids. |

**Description:** If your application uses one of the predefined menu ids, you can call this function to check whether you need to enable or disable the menu item.

**Return Value:** This function returns TRUE if the menu item should be enabled, otherwise it returns FALSE.

**See Also:**
HtsMenuSelect
HtsCommand

## HtsMenuSelect

**Return the 'check' status of a menu item.**

BOOL HtsMenuSelect(hWnd, MenuId)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int MenuId; | The menu id to return the status. Please refer to the 'HtsCommand' function for a list of available menu ids. |

**Description:** If your application uses one of the predefined menu ids, you can call this function to test whether you need to check the menu item.

**Return Value:** This function returns TRUE if the menu item should be checked, otherwise it returns FALSE.

**See Also:**
HtsMenuEnable
HtsCommand

## HtsModified

**Check if the current document is modified.**

BOOL HtsModified(hWnd)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |

**Return Value:** This function returns TRUE if the current document is modified. If the current document contains embedded frames, then this function returns TRUE if the text in any of the embedded frames is modified.

## HtsParaNormal

**Turn off the paragraph properties.**

BOOL HtsParaNormal(hWnd, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| BOOL repaint; | Repaint the screen after this operation. |

**Description:** This function turns off the following paragraph attributes: header, lists, blockquote, address, and preformatted text.

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsSetListEx
HtsSetMiscParaType

## HtsPositionName

**Position at a given tag name within the current document.**

BOOL HtsPositionName(hWnd, tag)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| LPBYTE tag; | The tag name to position at |

**Description:** This function is used to position at a given tag within the current document. The tag name is specified in an HTML document using the <A> element with a 'name=' attribute.

If the tag is located, the line containing the tag is displayed at the top of the window.

**Return Value:** This function returns TRUE when successful.

# HtsRead

**Read an HTML document or buffer into the specified TER window.**

BOOL HtsRead(hWnd, InputType, FileName, hBuf, BufLen, TagName)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int InputType; | Input type:<br>HTML_FILE: Specify the input file name using the FileName argument.<br>HTML_BUF: Specify the input buffer using the hBuf and BufLen arguments. |
| LPBYTE FileName; | Input file name. The file name may be prefixed by a subdirectory path. Set to NULL when the 'InputType' argument is HTML_BUF. |
| HGLOBAL hBuf; | Global memory handle to a buffer containing HTML text. Set to NULL when the 'InputType' argument is HTML_FILE. |
| long BufLen; | The size of the text specified in the 'hBuf' parameter. Set to NULL when the 'InputType' argument is HTML_FILE. |
| LPBYTE TagName; | The tag name to position at. A tag name is indicated in an HTML file using the <A> markup element with a 'name=' attribute.<br>Specify NULL to position at the top of the file. |

**Description:** This function is used to read a new document into the specified TER window. The data may be contained in a disk file or in a memory buffer. The data must be in the HTML format.

This function sends HTS_PICTURE message to the parent window whenever it encounters a picture element in the document.

**Return Value:** This function returns a TRUE value when successful.

**Example:**

```
// read an HTML disk file
ReadHts(hWnd,HTML_FILE, "myfile.htm",NULL,0,NULL);
// read an HTML disk file and position at a tag called
   'begin here'.
ReadHts(hWnd,HTML_FILE, "myfile.htm",NULL, 0, "begin here");
```

```
// read a memory buffer containing HTML text
HGLOBAL hBuf;
long BufLen;
ReadHts(hWnd,HTML_BUF,NULL,hBuf,BufLen,NULL);
```

## HtsReformat

**Save the document to a temporary buffer and reload.**

BOOL HtsReformat(hWnd)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |

**Description:** This function is useful for redisplaying a document.

**Return Value:** This function returns TRUE when successful.

## HtsSave

**Save the current HTML document to a file or to a buffer.**

BOOL HtsSave(hWnd, OutputTo, FileName, hBuf, BufLen)

| | | |
|---|---|---|
| HWND hWnd; | The handle of the window to be accessed | |
| int OutputTo; | Output to: | |
| | HTML_FILE: | Specify the output file name using the FileName argument. |
| | HTML_BUF: | The output buffer is returned via the hBuf and BufferLen parameters. |
| LPBYTE FileName; | Output file name. The file name may be prefixed by a subdirectory path. Set to NULL when the 'OutputTo' argument is HTML_BUF. | |
| HGLOBAL far *hBuf; | When saving to a buffer, this pointer receives the global memory handle of the buffer containing the text. | |
| LPLONG BufLen; | When saving to a buffer, this pointer receives the buffer length. | |

**Description:** This function is meant to be called from your application's save menu procedure to save the current document.

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsWrite
HtsClose

## HtsSetActiveWnd

**Set Active Window handle.**

BOOL HtsSetActiveWnd(hWnd)

HWND hWnd; The handle of the window to be accessed

**Description:** In a document containing embedded frames, the active window is automatically selected when the user click inside a frame window. This function can be used to override the current active frame window.

**Return Value:** This function returns TRUE if successful.

**See Also:**
HtsGetActiveWnd2

## HtsSetBkColor

**Set the background color for a new control.**

BOOL HtsSetBkColor(BkColor)

COLORREF BkColor; Background color

**Description:** This function is used to set a non-default background color for a control. This function must be called before the HtsInitailize function.

**Return Value:** This function always returns TRUE.

**See Also:**
HtsSetNewBkColor

## HtsSetBkPict

**Set the background picture file.**

BOOL HtsSetBkPict(hWnd, PictFile)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| LPBYTE PictFile; | Name of the picture file. Set to NULL to remove existing background picture. |

**Return Value:** This function returns TRUE when successful.

**See Also:**
HtsSetNewBkColor

## HtsSetDefaultTarget

**Set the default target the frame window.**

BOOL HtsSetFrameName(hWnd, target)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed. This window handle is used as given without any modification described in the beginning of this chapter. |
| LPBYTE target; | New default target. |

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsSetFrameName

## HtsSetDocTitle

**Set the document title.**

BOOL HtsSetTitle(hWnd, title)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| LPBYTE title; | docment title. Set to "" to remove the existing title. The document title is saved to the HTML file using the 'title' tag. |

**Return Value:** This function returns a TRUE value when successful.

## HtsSetDownloadDir

**Set the directory path to download and cache internet files.**

BOOL HtsSetDownloadDir(DirPath)

LPBYTE DirPath; The directory path to save the internet files

**Return Value:** This function returns TRUE when successful, otherwise it returns a FALSE value.

**See Also:**
HtsIsLoaclFile
HtsIsHttpFile
HtsInternetGet

## HtsSetFlags

**Set the operating flags.**

DWORD HtsSetFlags(hWnd, SetReset, flags)

HWND hWnd; The handle of the window to be accessed

BOOL SetReset; TRUE to set the given flags, FALSE to reset them.

DWORD flags; Flags to set or reset. Current the following flags are available:

| | |
|---|---|
| HFLAG_OUTPUT_DEF_FONT: | Write the default font typeface to the output file. |
| HFLAG_OUTPUT_DEF_FONTSIZE: | Write the default font size to the output file. |
| HFLAG_OUTPUT_DEF_FONTCOLOR: | Writethe default font color to the output file |
| HFLAG_NO_TAG_FONT: | Do not write the default font information for the tags. |
| HFLAG_SAVE_CELL_WIDTH: | Write the default cell width to the output file. |
| HFLAG_NO_PICT_PATH: | Do not save the picture path in the output file. |

| | |
|---|---|
| HFLAG_NO_FORM_SHADING: | Do not shade the form area in the edit mode. |
| HFLAG_NO_TOGGLE_MSG: | Do not show the warning message when toggling from the edit to the view mode. |
| HFLAG_FIXED_VSCROLL: | Do not automatically hide or display the vertical scroll bar. |
| HFLAG_SAVE_PICT_AS_WMF: | During RTF to HTML conversion save the embedded pictures as metafiles. By default, the embedded files are stored in the PNG format. |
| HFLAG_NO_SCRIPT: | Suppress script processing. |
| HFLAG_ROUND_BULLET: | Use the round bullet for every list level. |
| HFLAG_NO_DIV: | Use the <p> tag instead of the  tag for HTML output. |
| HFLAG_NO_SPAN_TAG: | Do not output the  tag. |
| HFLAG_WRITE_LINK_STYLE: | Write the link style and color to the output file. |
| HFLAG_NO_INTERNET: | Do not access the internet files. |
| HFLAG_NO_HEAD: | Do not write the 'head' section |
| HFLAG_NO_BODY: | Do not write the 'body' tag |
| HFLAG_NO_FONT: | Do not write the 'font' tag |
| HFLAG_NO_STYLE: | Do not write the character style tags |
| HFLAG_NO_LIST_MARGIN: | Do not write the margin information for the "<li>" tag. |
| HFLAG_NO_XLATE_LINK: | Do not translate % characters |

**Return Value:** This function returns the current flag values.

## HtsSetFlags2

**Set the operating flags.**

DWORD HtsSetFlags2(hWnd, SetReset, flags)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| BOOL SetReset; | TRUE to set the given flags, FALSE to reset them. |
| DWORD flags; | Flags to set or reset. Current the following flags are available: |

| | |
|---|---|
| HFLAG2_DOCTYPE: | Generate DOCTYPE information in the document header. |

**Return Value:** This function returns the current flag values.

## HtsSetFont

**Set the new font information for an HTML style element.**

BOOL HtsSetFont(hWnd, font, typeface, PointSize, style, TextColor, TextBkColor, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int font; | Font id for an HTML style element. Refer to the 'HtsGetFont' API for a list of the font ids. |
| LPBYTE typeface; | The pointer to receive the font typeface. This buffer should be able to hold upto 32 characters. |
| int PointSize; | The new point size for the font. |
| UINT style; | The new style bits for the font: |

| | |
|---|---|
| BOLD: | Bold |
| ITALIC: | Italic |
| ULINE: | Underline |
| STRIKE | Strikethrough |

| | |
|---|---|
| COLORREF TextColor; | The new text foreground color. |

| | |
|---|---|
| COLORREF TextBkColor; | The new text background color. |
| BOOL repaint; | TRUE to refresh the window after applying the new text attributes. |

**Description:** This function is used to set the new font specification for an HTML style element. The new font attributes are specified using the function parameters. If you wish to retain the old value for a parameter, use the 'HtsGetFont' function to retrieve the existing value, and then use the 'HtsSetFont' to specify the existing values for the parameters that you do not wish to change.

**Return Value:** This function returns a TRUE value when successful.

**Example:**

```
BYTE typeface[32];
int PointSize;
UINT style;
COLORREF TextColor, TextBkColor;
HtsGetFont(hWnd, FONT_H2, typeface, &PointSize, &style,
          &TextColor, &TextBkColor);
HtsSetFont(hWnd,FONT_H2, "Arial", PointSize, style, 0xFF,
          TextBkColor);
// The above example modifies the typeface and text color
   for the 'Header 2' HMTL style element.  The PointSize,
   style and text background are left unchanged.
```

**See Also:**
HtsSetFontDlg
HtsGetFont
HtsSetLinkInfo

## HtsSetFontDlg

**Allow the user to edit the font information for an HTML style element using a dialog box.**

BOOL HtsSetFontDlg(hWnd, font)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int font; | Font id for an HTML style element. Refer to the 'HtsGetFont' API for a list of the font ids. You can set this parameters to -1 to let the user select an HTML style element from a list of style elements. |

**Description:** This function allows the user to modify the font information for a desired HTML style element.

**Return Value:** This function returns a TRUE value when successful.

**Example:**

```
HtsSetFontDlg(hWnd, -1);
```

**See Also:**
HtsGetFont
HtsSetLinkInfo

## HtsSetFontSize

**Set the font size.**

BOOL HtsSetFontSize(hWnd, size, relative, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int size; | If the 'relative' parameter is TRUE, the size must be between -7 and 7. If the 'relative' parameter is FALSE, the size must be between 1 and 7. To invoke a user dialog box, set the size to 0. |
| BOOL relative; | Set to TRUE to change the font size relatively. |
| BOOL repaint; | TRUE to refresh the window after this operation. |

**Description:** The 'size' parameter is combined with the 'base font size' for the document to derive a number between 1 and 7. The higher number indicates bigger fonts.

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsSetFontDlg
HtsGetFont
HtsSetLinkInfo
HtsSetFontSizeTbl

## HtsSetFontSizeTbl

**Set the font size table.**

BOOL HtsSetFontSizeTbl(size1, size2, size3, size4, size5, size6 size7)

| | |
|---|---|
| int size1; | The pointsize for HTML font size #1 |

| | |
|---|---|
| int size2; | The pointsize for HTML font size #2 |
| int size3; | The pointsize for HTML font size #3 |
| int size4; | The pointsize for HTML font size #4 |
| int size5; | The pointsize for HTML font size #5 |
| int size6; | The pointsize for HTML font size #6 |
| int size7; | The pointsize for HTML font size #7 |

**Description:** This function overrides the default point sizes for the HTML font sizes. If you wish to override the default values, call this function *before any editor window is opened.*

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsSetFontSize

## HtsSetForm

**Set the Form attribute for a paragraph.**

BOOL HtsSetForm(hWnd, set, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| BOOL apply; | Set to TRUE to turn a paragraph into a form. Set to FALSE to reset this attribute. The input fields can be inserted only within form paragraph. |
| BOOL repaint; | Repaint the screen after this operation. |

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsEditFormId
HtsSetFormId

## HtsSetFormId

**Set a form id for a form.**

BOOL HtsSetFormId(hWnd, id)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| BOOL id; | Set a form id for the current form. |

**Description:** Before this function is called, the cursor must be located on a form paragraph. The current form must also have at least one input field to be able to associate a form id with it.

**Return Value:** This function returns a TRUE value when successful.

**See Also:** HtsEditFormId, HtsSetForm

**See Also:**
HtsEditFormId

## HtsSetFrameName

**Set a new name for a frame window.**

BOOL HtsSetFrameName(hWnd, name)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed. This window handle is used as given without any modification described in the beginning of this chapter. |
| LPBYTE name; | New frame name. |

**Return Value**: This function returns a TRUE value when successful.

**See Also:**
HtsGetFrameName

## HtsSetHeader

**Apply the header property.**

BOOL HtsSetHeader(hWnd, level, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int level; | Header level (1 to 7) |
| BOOL repaint; | Repaint the screen after this operation. |

**Description:** This function turns the current paragraph into a header paragraph of the given level. Call this function with the 'level' argument set to 0 to turn off the header property.

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsParaNormal
HtsSetListEx
HtsSetMiscParaType

## HtsSetLinkInfo

**Set new text color and style for the link text.**

BOOL HtsSetLinkInfo(hWnd, style, TextColor)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| UINT style; | The new style bits for the font: |
| | BOLD: Bold |
| | ITALIC: Italic |
| | ULINE: Underline |
| | STRIKE Strikethrough |
| COLORREF TextColor; | The new link foreground color. |

**Description:** This function is used to set new text style and color for the link text. If you wish to edit only one of the parameter, you can use the 'HtsGetLinkInfo' API to get the existing information. Specify the existing value for the parameter that you do not wish to modify.

**Return Value:** This function returns a TRUE value when successful.

**Example:**

```
UINT style;
COLORREF TextColor;
HtsGetLinkInfo(hWnd, &style, &TextColor);
HtsSetLinkInfo(hWnd, style, 0xFF);
// The above example modifies the color for the link text.
   The style is left unchanged.
```

**See Also:**
HtsGetLinkInfo

HtsSetLinkInfoDlg
HtsSetFont

## HtsSetLinkInfoDlg

**Allow the user to edit the color and style for the link text.**

BOOL HtsSetLinkInfoDlg(hWnd)

HWND hWnd; The handle of the window to be accessed

**Description:** This function allows the user to modify the color and style for the link text.

**Return Value:** This function returns a TRUE value when successful.

**Example:**

```
HtsSetLinkInfoDlg(hWnd);
```

**See Also:**
HtsSetLinkInfo
HtsSetFontDlg

## HtsSetListEx

**Apply the paragraph list property.**

BOOL HtsSetList(hWnd, type, level, DtDd, repaint)

BOOL HtsSetListEx(hWnd, type, level, DtDd, NbrSymbol, repaint)

**This function is now deprecated. Please use the functions available in TE Edit Control to use the List functionality.**

HWND hWnd; The handle of the window to be accessed

int type; The list type can be one of the following:

PARA_LIST_OL: Ordered list

PARA_LIST_UL: Unordered list

PARA_LIST_DL: Definition list

Also, you can set the 'type' to 0 to leave it unchanged.

int level; List level (1 to 7). Also, you can set the 'level' to 0 to leave it

unchanged.

int DtDd; — Set to TERM_DT to indicate the definition term, set to TERM_DD to indicate the definition data, set to 0 to leave it unchanged. This argument is used only when the 'type' is set to PARA_LIST_DL.

int NbrSymbol; — For an 'ordered' list, this parameters specifies the type of number symbol to use:

| | |
|---|---|
| NBR_DEC: | Decimal numbering |
| NBR_UPR_ALPHA: | Number using uppercase alphabetic characters. |
| NBR_LWR_ALPHA: | Number using lowercase alphabetic characters. |

The above constants are defined in ter.h, ter.bas or ter_prot.pas files.

BOOL repaint; — Repaint the screen after this operation.

**Description:** If both the 'type' and the 'level' arguments are set to 0, then this function will toggle the 'new item' property of the current list item. A new list item is bulleted or numbered, whereas a paragraph without this property indicates a next paragraph of the current list item.

Please note that the numbering for the ordered list is not available in the edit mode.

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsParaNormal
HtsSetHeader
HtsSetMiscParaType

## HtsSetMiscCharType

**Set Example, Keyboard, Sample, Variable, Typewriter, TR or HV styles.**

BOOL HtsSetMiscCharType(hWnd, type, repaint)

HWND hWnd; — The handle of the window to be accessed

int type; — The type can be one of the following:

| | |
|---|---|
| FONT_EXAMPLE: | Example |

| | | |
|---|---|---|
| | FONT_KBD: | Keyboard |
| | FONT_SAMP: | Sample |
| | FONT_VAR: | Variable |
| | FONT_TT: | Typewriter |
| | FONT_TR: | Times Roman |
| | FONT_HV: | Helvetica |
| | You can also set the 'type' argument to 0 to reset all miscellaneous character styles. | |
| BOOL repaint; | Repaint the screen after this operation. | |

## HtsSetMiscParaType

**Set Blockquote, Address, or Preformatted text attributes.**

BOOL HtsSetMiscParaType(hWnd, type, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int type; | The type can be one of the following: |
| | PARA_BLOCKQUOTE |
| | PARA_ADDRESS |
| | PARA_PRE |
| BOOL repaint; | Repaint the screen after this operation. |

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsParaNormal

## HtsSetNewBkColor

**Set the background color for the current file.**

BOOL HtsSetNewBkColor(hWnd, color, set)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| COLORREF color; | New background color |
| BOOL set; | TRUE to set the specified color, FALSE to remove any existing background color. |

**Return Value:** This function returns TRUE when successful.

**See Also:**
HtsSetBkPict
HtsSetNewTextColor

## HtsSetNewTextColor

**Set the foreground color for the current file.**

BOOL HtsSetNewTextColor(hWnd, color)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| COLORREF color; | New text foreground color. When the document is saved, this color is written out in the "body" tag. |

**Return Value:** This function returns TRUE when successful.

**See Also:**
HtsSetNewBkColor

## HtsSetParaSpace

**Toggle space before and after the paragraph.**

BOOL HtsSetParaSpace(hWnd, SetSpace, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| BOOL SetSpace; | Set to TRUE to create space before and after the paragraph. Set to FALSE to remove such space. |
| BOOL repaint; | Repaint the screen after this operation. |

**Return Value:** This function returns a TRUE value when successful.

## HtsSetPicture

**Replace a picture with a new picture.**

BOOL HtsSetPicture(hWnd, PictId, PictType, PictFile)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int PictId; | The picture id of the picture to replace. Your application is informed of the picture for a picture using the HTS_PICT_ID message. |
| int PictType; | Picture type of the new picture file. Please refer to the description of HTS_PICTURE message (chapter: Message Communication) for a list of valid picture types. |
| LPSTR PictFile; | The pathname of the new picture file. |

**Description:** This function is useful for replacing a placeholder picture with the real picture in a document.

During the initial document load time, your application receives the HTS_PICTURE message to resolve the pathname of a picture in the document. When such a picture is located on a remote network, your application might return a pathname of a local placeholder picture file in response to the HTS_PICTURE message. The HTS_PICTURE message is followed by the HTS_PICT_ID message, which informs your application about the picture id of the preceding picture. After the document is completely loaded, the control displays the document with the placeholder pictures. Your application can then retrieve the real picture from the remote source. The original placeholder picture can then be replaced by the real picture using this function.

**Return Value:** This function returns a TRUE value when successful.

**Example:**

```
HtsSetPicture(hWnd, PictId, PICT_BMP, "new.bmp");
```

**See Also:**
HtsSetBkPict
HtsSetNewTextColor

## HtsSetPictUseMap

**Set the 'usemap' value for a picture.**

BOOL HtsSetPictUseMap(hWnd, PictId, UseMap)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int PictId; | The picture id of the picture. |
| LPBYTE UseMap; | UseMap value string. |

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsInsertPicture

## HtsSetPictFileBase

**Set the the base name for the picture file during RTF to HTML conversion.**

BOOL HtsSetPictFileBase(FileBase)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| LPBYTE FileBase; | File name base for the picture files. The default file name base is "HTS_". |

**Return Value:** This function returns a TRUE value when successful.

## HtsSetReadOnly

**Set or reset the read only status for the current document.**

BOOL HtsSetReadOnly(hWnd)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |

**Description:** If the current document contains embedded frames, then this function applies the new read-only status to all frame windows in the document.

**Return Value:** This function returns the previous read-only status for the document.

## HtsSetTableWidth

**Set the current table width.**

BOOL HtsSetTableWidth(hWnd, width, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int width; | The table width in pixels. You can also set a percentage width by specifying a negative value, i.e. set 'width' to -100 to specify 100 percent width. |
| BOOL repaint; | TRUE to repaint the screen after this operation. |

**Return Value:** This function returns a TRUE value when successful.

## HtsSetTarget

**Set the target frame for the hyperlink jump.**

BOOL HtsSetTarget(hWnd, target)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| LPBYTE target; | target frame name. |

**Description:** When the user clicks on a hyperlink text within a framed document, the editor automatically sets the target frame for the hyperlink jump. This target remains effective until the next HtsRead function is called to load the new document. In a typical implementation, your application will call the HtsClearWindow and HtsRead functions when it receives the HTS_LINK message. If your implementation does not call the HtsRead function, then it must call the HtsSetTarget function to reset the target as follows:

```
HtsSetTarget(hWnd,"");
```

This function can also be used to override the target frame as specified in the 'href' tag for the current hyperlink.

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsGetTarget

## HtsSetUserTag

**Set the text data for a user tag.**

BOOL HtsGetUserTag(hWnd, TagId, TagText)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| int TagId; | Tag id for the user tag to set |
| LPBYTE TagText; | The tag text. Set this parameter to NULL or "" to delete the user tag. |

**Description:** This function sets the text data associated with the given user tag id. HTML add-on saves the user tag id and the associated tag text in the 'head' section of the HTML file using the 'meta' tag:

```
<meta UserTagId:nn UserTagData="tag data">
```

**Return Value:** This function returns TRUE when successful.

**See Also:**
HtsGetUserTag

## HtsString2Lparam

**Copy text from a string variable to a long parameter (lParam)**

BOOL HtsString2Lparam(string,lParam)

| | |
|---|---|
| LPBYTE string; | Source text pointer |
| long lParam; | Destination pointer in the long form. |

**Description:** This function is used by a Visual Basic application to copy the text from a string variable to a long parameter.

**Return Value:** This function always returns a TRUE value.

**Example:**

```
sub html(message as Integer,wParam as Integer,lParam as Long)
  Dim text as string
  text=space$(300)    ' allocate 100 bytes for the text
                        variable
  HtslParam2String(lParam,text)  ' copy the text from
                                   lParam to text
  text="new text"
  HtsString2Lparam(text,lParam) ' return the new string
end sub
```

**See Also:**
HtsLparam2String

## HtsUpdateLinkEx

**Update the 'href' and 'name' parameters for the link at the cursor position.**

BOOL HtsUpdateLink(hWnd, href, name, repaint)

BOOL HtsUpdateLinkEx(hWnd, href, name, target, repaint)

| | |
|---|---|
| HWND hWnd; | The handle of the window to be accessed |
| LPBYTE href; | The new 'href' parameter. |
| LPBYTE name; | The new 'name' parameter. |
| LPBYTE target; | The new frame target for the link. |
| BOOL repaint; | TRUE to repaint the screen after this operation. |

**Return Value**: This function returns a TRUE value when successful.

**See Also:**
HtsGetLinkDataEx

## HtsWrite

**Write the current HTML document to a file or to a buffer.**

BOOL HtsWrite(hWnd, OutputTo, FileName, hBuf, BufLen,WriteSelection)

| | | |
|---|---|---|
| HWND hWnd; | The handle of the window to be accessed | |
| int OutputTo; | Output to: | |
| | HTML_FILE: | Specify the output file name using the FileName argument. |
| | HTML_FILE_DLG: | Invokes a dialog box which allows the user to select a file name to save. |
| | HTML_BUF: | The output buffer is returned via the hBuf and BufferLen parameters. |

| | |
|---|---|
| LPBYTE FileName; | Output file name. The file name may be prefixed by a subdirectory path. Set to NULL when the 'OutputTo' argument is HTML_BUF. |
| HGLOBAL far *hBuf; | When saving to a buffer, this pointer receives the global memory handle of the buffer containing the text. |
| LPLONG BufLen; | When saving to a buffer, this pointer receives the buffer length. |
| BOOL WriteSelection; | Set this flag to TRUE if you wish to save only the selected text. |

**Description:** This function is used to write the current document to a file or a buffer.

**Return Value:** This function returns a TRUE value when successful.

**See Also:**
HtsSave

## TvbSetEventResult

**Return a result in an event handler.**

BOOL TvbSetEventResult(result)

| | |
|---|---|
| long result; | result to return to the event handler |

**Description:** This function is used by a Visual Basic or Delphi application to return the result to an event handler.

**Return Value:** This function always returns a TRUE value.

**Example:**

```
sub html(message as Integer,wParam as Integer,lParam as long)
  Dim PictType as Integer
  if (message = HTML_PICTURE) then
    ...
    ...
    ...
    PictType = PICT_BMP;
    TvbSetEventResult(PictType);    // return picture type
  endif
end sub
```

# Viewer/Editor Modes

The selection of viewer or editor mode is controlled by the read-only status of the TER control. To invoke the HTML viewer, turn on the read-only mode before reading the document. You can specify the read-only mode either during the creation of the control or by using the TerSetReadOnly API (exported by the TER control). If the read-only mode is not selected, the document is opened in the edit mode. In the edit mode, you can access additional APIs to edit the document.

The appearance of the document in the edit mode have these variations from the viewer mode:

The form text is displayed in a darker shade. This helps the user visually to identify the beginning and end of the form.
The numbering for the ordered list is disabled. Instead of a list item number, a bullet is displayed.
During the process of editing, the paragraph spacing and the table size may change. At any time to view the actual formatting, select the reformat option (or HtsReformat API) from the menu.

# Message Communication

The Hts dll sends the following messages to the parent window of the editor:

**HTS_BASE:**

The Hts dll sends this message when it encounters the 'base' HTML marker.

| | |
|---|---|
| wParam | The window handle of the editor. |
| lParam | The pointer to the base URL string. |

Return Value: The dll ignores the return value from this message.

**HTS_BK_PICT:**

The Hts dll sends this message when it encounters the 'body' tag with background picture specification. This message follows immediately before the HTS_PICTURE message which is sent to resolve the picture file name for the background picture..

| | |
|---|---|
| wParam | The window handle of the editor. |
| lParam | This parameter is not used. |

*Return Value:* The dll ignores the return value from this message.

**HTS_FORM:**

The Hts dll fires this message when the user submits a form by clicking on a 'submit' button, or form image, or a single 'one line text' box.

| | |
|---|---|
| wParam: | The window handle of the editor. |
| lParam: | The form id of the form being submitted. |

*Application Response:* Upon receiving this message, your application will typically retrieve the form information by using the HtsGetFormInfo function. This function returns the total number of fields in the form. Your application can then use the HtsGetFieldData function to retrieve the user entered data for each field in the form.

*Return Value:* The dll ignores the return value from this message.

**HTS_FRAME_FILE**:

The dll may send this message to the parent window before reading a document into a frame of a frame-set document.

| | |
|---|---|
| wParam | The window handle of the editor. |
| lParam | The pointer to the file URL. |

*Application Response:* Your application will resolve the URL into a MSDOS file path name. It may need to retrieve the file from a remote location or may have it available in the local machine. In either case, your application writes the file path name in the pointer provided by the lParam parameter. The file pathname must be in the MSDOS format. If this message is not processed, then the dll attempts to read the file specified by the original URL.

*Return Value:* Not used.

**HTS_FRAME_NAME**:

The dll may send this message to the parent window before reading a document into a frame of a frame-set document.

| | |
|---|---|
| wParam | The window handle of the editor. |
| lParam | The pointer to the frame name. |

*Application Response:* This message is sent before sending the HTS_FRAME_FILE message and it denotes the frame into which the next frame file is being read. This message is for information only. Once the entire frame set is loaded, your application can use this frame name with the HtsGetActiveWnd2 function to get the window handle of the frame.

*Return Value:* Not used.

**HTS_LINK:**

The Hts dll *posts* this message (in the viewer mode only) when it needs your application to load a new document or respond to a click at a particular location on a hot spot image.

| | |
|---|---|
| wParam | The window handle of the editor. |
| lParam | The pointer to the destination path. The destination path may be a file name in the current directory or a complete URL. The path name may also be suffixed with either the destination tag name or a mouse click location on a hot spot image.<br><br>The tag name is delimited from the URL string using '#' character. Example:<br><br>`//localhost/c:\mydir\myfile.htm#mytag`<br><br>When a hot spot image has an attribute of 'ISMAP', the URL is suffixed with a mouse location string in the format of ?X,Y. Example:<br><br>`//localhost/c:\mydir\myfile.htm?10,15`<br><br>In the above example the mouse click was record on x = 10 and y=15 pixel position. The coordinates are relative to the top, left corner of the picture. |

*Application Response:* In response to this message your application will typically clear the existing document from the window and read the new document specified by the URL string. If a destination tag is specified, your application will position the document at the specified tag. Your application can use the HtsClearWindow and HtsRead functions for clearing the window and reading a new document. Your application may also interpret the mouse location as desired.

*Return Value:* The dll ignores the return value from this message.

**HTS_LINK_INFO:**

This message is similar to HTS_LINK, except that this message is sent in the 'edit' mode. This message is sent for information only.

**HTS_INT_LINK:**

This message is similar to HTS_LINK, except that this message is sent for an internal jump. This message is sent only in the 'ReadOnly' mode. This message is send for information only. The editor automatically performs internal jumps.

**HTS_LINK_URL:**

The Hts dll sends this message during the initial read process when it encounters an anchor link. This message informs your application of the URL used for the link. Your application does not need to respond to this message as the DLL sends the application

the HTS_LINK message when the user actually activates a link.

| | |
|---|---|
| wParam | The window handle of the editor. |
| lParam | The pointer to the current anchor URL string. |

**Return Value:** The dll ignores the return value from this message

**HTS_PICT_ID**:

The dll sends this message to the parent window when reading a new document into the editor window. This message is followed immediately after the HTS_PICTURE message. It informs your application about the picture id of the preceding picture.

| | |
|---|---|
| wParam | The window handle of the editor. |
| lParam | The picture id of the preceding picture. |

*Application Response:* Your application will typically ignore this message unless the previous picture was a placeholder picture to be later replaced by a real picture. If the previous picture was a placeholder, your application saves the picture id of that picture to be later used for replacing the original placeholder picture. For more information, please refer to the HtsSetPicture function.

*Return Value:* Ignored.

**HTS_PICTURE:**

The dll may send this message to the parent window when reading a new document into the editor window. During the parsing process when the editor encounters an image, it informs your application about the URL of the image.

| | |
|---|---|
| wParam | The window handle of the editor. |
| lParam | The pointer to the image URL. |

Application Response: Your application will resolve the URL into a MSDOS file path name. It may need to retrieve the picture from a remote location or may have it available in the local machine. In either case, your application writes the picture path name in the pointer provided by the lParam parameter. The picture pathname must be in the MSDOS format. Your application also returns an integer value designating the picture type:

| | |
|---|---|
| PICT_NONE: | Picture not available. The editor displays the alternate text string in place of the picture. |
| PICT_BMP: | Windows device independent bitmap format. |
| PICT_WMF: | Windows placeable metafile format. |
| PICT_JPEG: | Jpeg format. |

PICT_PNG: Portable Network graphic format.

In addition, Sub Systems offers a free support for GIF (PICT_GIF) and TIFF (PICT_TIFF) formats when you have a Unisys license to use thse formats.

Example: If the lParam was set to

```
//localhost/c:\mydir\mypict.bmp
```

your application will set lParam to c:\mydir\mypict.bmp:

```
lstrcpy((LPSTR)lParam, "c:\mydir\mypict.bmp");
```

and return PICT_BMP.

*Return Value:* Picture type as described above.

**HTS_SAVE_PICTURE**:

The dll may send this message to the parent window when saving a picture file information. This message informs your application about the URL of the image and lets you modify the url if you desire.

| | |
|---|---|
| wParam | The window handle of the editor. |
| lParam | The pointer to the image URL. The modified picture URL can also be passed using this pointer. |

*Return Value:* Not used.

**HTS_TITLE**:

The editor sends this message when it encounters the 'title' control element in the HTML document.

| | |
|---|---|
| wParam | The window handle of the editor. |
| lParam | The pointer to the title string. |

*Application Response:* Your application will typically display the title string as the window caption. Example:

```
SetWindowText((HWND)wParam,(LPSTR)lParam);
```

*Return Value:* The editor ignores any return value from this message.

# Interaction Between Hts and TER DLLs

The Hts dll and the TER dll work together to make an HTML viewer control.

**The TER dll performs the following tasks:**

Display the text in a window.

Provide user interface for scrolling.
Capture mouse clicks on hyperlink areas and dispatch the link messages to the Hts dll.

**The Hts dll performs the following tasks:**

Parse the HTML document and insert the text into the TER dll. The text is inserted as protected text.
Translate the image formats not supported by the TER dll.
Respond to the mouse messages from the TER dll.

On initialization the Hts dll modifies the operating parameters for the TER dll. The initialization of the TER dll consists of the following tasks.

The Hts dll requests the TER dll to reserve a block of global memory for the internal use of the Hts dll. This memory is automatically freed up by the TER dll before exiting.
Create fonts for each HMTL style elements.
Set the tab width, background color and link text style. The editor is also programmed not to change color of the protected text, issue link message on single mouse clicks and turn off caret display over protected text.
The Hts dll also registers a callback function with the TER dll to receive the hyperlink message

.

# Visual Basic Interface

The HTML Add-on can be used with the toc31.ocx (available for Win32 only) which is included with TE Edit control. The Hts dll interacts with the TOC ocx by firing the 'HTML' event. The Hts dll also interacts with the TER dll by calling the TER dll functions. You will need the following dll and ocx files in your working directory:

| | |
|---|---|
| ter29.dll | TE Edit control main dll |
| hts26.dll | HTML Add-on dll |
| toc31.ocx | OCX interface for ter29.dll |

Also, your application needs to include the following files in your project:

| | |
|---|---|
| Hts.BAS | Function declarations for the Hts dll |
| TER.BAS | Function declarations for the TER dll |

Please refer to the TE Edit control manual for the instructions to create a TER control in your application.

The ReadOnly flag determines the viewer or editor session. Turn on the ReadOnly flag to create an HTML viewer.

**Loading the First Document**: To load the first HTML document, call these two Hts functions when loading the form:

```
HtsInitialize
HtsRead
```

Please refer to the *Application Interface Functions* for a detailed description of these functions.

**HTML Interface**: The Hts dll interacts with your application using the 'Html' event. This event is fired to pass the HTML messages to your application. It takes these 3 parameters:

```
message as Long
wParam as Long
lParam as Long
```

These three parameters for each Hts message are described in the Message Communication chapter. Your application needs to respond to these messages. The package includes a Visual Basic demo program (DMO_OCX) which can be used as a reference for writing your application response to these messages.

The HTML Add-on can be used with the toc31.ocx (available for Win32 only) which is included with TE Edit control. The Hts dll interacts with the TOC ocx by firing the 'HTML' event. The Hts dll also interacts with the TER dll functions. You will need the following dll and ocx files in your working directory:

| | |
|---|---|
| ter29.dll | TE Edit control main dll |
| hts26.dll | HTML Add-on dll |
| toc31.ocx | OCX interface for ter29.dll |

Also, your application needs to include the following files in your project:

| | |
|---|---|
| Hts.BAS | Function declarations for the Hts32 dll |
| TER.BAS | Function declarations for the TER32 dll |

Please refer to the TE Edit control manual for the instructions to create a TER control in your application.

The ReadOnly flag determines the viewer or editor session. Turn on the ReadOnly flag to create an HTML viewer.

**Loading the First Document**: To load the first HTML document, call these two Hts functions when loading the form:

```
HtsInitialize
HtsRead
```

Please refer to the *Application Interface Functions* for a detailed description of these functions.

**HTML Interface**: The Hts dll interacts with your application using the 'Html' event. This event is fired to pass the HTML messages to your application. It takes these 3 parameters:

```
message as Long
wParam as Long
lParam as Long
```

These three parameters for each Hts message are described in the Message Communication chapter. Your application needs to respond to these messages. The package includes a Visual Basic demo program (DMO_OCX) which can be used as a reference for writing your application response to these messages.

# Recompile the DLL

If you need to modify the DLL source code and recompile within the Visual C++ environment, follow these steps to create a Visual C++ project:

**Files:** Hts*.C, Hts.DEF and Hts.RC

**Executable Type:** Windows DLL

**Alignment (Compiler Option):** 1 Byte

Remaining parameters should be left at their default values.