Software License Agreement

HTML Add-on For .NET Core

and

HTML Add-on For .NET Framework

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.

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 is product is add-on product for TE Edit Control. It provides for import/export of HTML text from the Editor.

Just as TE Edit Control, this product includes two dlls, hts26.dll, and Hss25.dll. The later is used for server operations.

The .NET Framework is backward compatible to .NET 2.

The .NET Core works with .NET 6 and later versions.

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, and make files that are necessary to incorporate the HTS routine into your application. In addition, you will also need the latest version of tern31.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.cs and dmo_*.cs files	Source code for the demo program
demo.exe	Executable demo program
demo.csproj	The project file to compile the demo.

htn24.DLL Source Files:

htn.cs	Main source module
htn*.cs	Other classes used by the htn.cs module.
htndlg_*.cs	Dialog boxes used by the product.

AssemblyInfo.cs	Assembly information file
htn.csproj	Project file which can be used to recompile the dll.

Visual Basic Interface and Demo Files (Not applicable to .NET Core version)

Form1.vb	vb source file
dmo_vbn.vbproject	The project file for the visual basic demo program.
AssemblyInfo.vb	Assembly information file for the demo program.

Make Files:

MAKE-MC.BAT	Compiles and links the HTS modules using
MAKE-MC	Microsoft c# compiler.

.Net Packages:

hts.26.0.0.0.nupkg: Nuget package for the desktop version of the product

hss.26.0.0.0.nupkg: Nuget package for the server version of the product

License Key

Your license key is e-mailed to you after your order is processed. You would set the license key using the HtsSetLicenseKey static function. This should be preferably done before creating the Htn object to avoid pop-up nag screens.

Htn.HtsSetLicenseKey("xxxxx-yyyyy-zzzzz")

Replace the 'xxxxx-yyyyy-zzzzz' by your license key.

When using HTML Add-on to simply export and import HTML data directly in the TE editor, you do not need to create the Htn object. Therefore, please use the TerSetHtnLicenseKey method available in the Tern class to set the license key for HTML Add-on:

tern1.TerSetHtnLicenseKey("xxxxx-yyyyy-zzzzz")

This method should be called after creating the Tern object.

Using HTML Add-on as a Viewer/Editor

Net Core Note: Include hts26.0.0.0.nuget or the latest version of the package in your product.

First create a TE editor window and then create an object of the Htn class. Please refer to the help file on TE Edit control for a discussion on how to create an object of the Tern class. Once an object of Tern class is created, you can create an Htn object as following:

```
using SubSystems.TE; // namespace containing TE methods - C# syntax
using SubSystems.HT; // namespace containing Htn methods

Imports SubSystems.TE   ' VBN syntax
Imports SubSystems.HT   ' VBN syntax

HTN htn1=new Htn(tern1)  // pass a Tern class object to bind to the
                            new Htn object
```

// set the license key. *Your license key is e-mailed to you after your order is processed.*

```
Htn.HtsSetLicenseKey(xxxx-yyyy-zzzz)
```

Net Framework only:

```
//To use the HTML Add-on menu system with the current form:

this.Menu=htn1.BuildMenu()   // here 'this' refers to the current form
                            // containing the Tern object
// Namespace:  The constants are available in the class hc within the
SubSystems.HT namespace. Example:

htn1.HtsSetFlags(True, hc.HFLAG_NO_PICT_PATH)
```

HTML Export/Import using TE

You do not necessarily need to create a Htn object to do standard import and export of HTML data within TE Edit control. Simply copy htn24.dll to the application directory containing tern31.dll.

Now use the TerSetHtnLicenseKey method to set the license key for HTML Add-on. *Your license key for HTML Add-on is e-mailed to you after your order is processed. Please note that your license key for TE Edit Control is not valid for HTML Add-on.*

```
tern1.TerSetHtnLicenseKey("xxxxx-yyyyy-zzzzz")
```

Export TE document as HTML:

First set the output format to SAVE_HTML using the TerSetOutputFormat method:

```
tern1.TerSetOutputFormat(tc.SAVE_HTML)
```

Now you can use any of the TE's output methods or properties to extract the data in the HTML format:

```
string HtmlText=tern1.Data
```

Import an HTML document into TE:

First set the input format as html:

```
tern1.TerSetFlags4(True,tc.TFLAG4_HTML_INPUT)
```

Now you can use any of the Te's input methods or properties to insert html data into TE:

```
tern1.Data = HtmlText
```

In the above examples, TE automatically creates an HTML object for you to do input or output of html data. However, you can also create the HTML object explicitly. The later method is suitable if you need to also call the html methods (such as HtsSetFlags) prior to doing export or import of html data.

Explicitly creating Htn object for doing HTML import/export:

First create an Htn object and pass it to TE:

```
using SubSystems.TE; // namespace containing TE methods - C# syntax
using SubSystems.HT; // namespace containing Htn methods

Imports SubSystems.TE   ' VBN syntax
Imports SubSystems.HT   ' VBN syntax

HTN htn1=new Htn(tern1)  // pass the Tern class object to bind to the
                          new Htn object
```

// set the license key. *Your license key for HTML Add-on is e-mailed to you after your order is processed. Please note that your license key for TE Edit Control is not valid for HTML Add-on.*

```
Htn.HtsSetLicenseKey(xxxx-yyyy-zzzz)
```

// Now pass the Htn object to TE.

```
tern1.TerSetHtnObject(htn1)
```

Now TE can use this Htn object to do import or export of HTML data as described earlier in this topic.

Methods

The 'Htn' class is defined in the *SubSystems.HT* name space.

The constants used by the editor methods are declared in the 'hc' class.

Before a Htn object is created, please set the license key to unlock the full functionality of the product.

The following is a description of the Htn methods 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

HtsSetFont
HtsSetFontDlg
HtsSetFontSize
HtsSetFontSizeTbl
HtsSetForm
HtsSetFormId
HtsSetFrameName
HtsSetHeader
HtsSetLinkInfo
HtsSetLinkInfoDlg
HtsSetListEx
HtsSetMiscParaType
HtsSetNewBkColor
HtsSetNewTextColor
HtsSetParaSpace
HtsSetPicture
HtsSetPictFileBase
HtsSetReadOnly
HtsSetTableWidth
HtsSetTarget
HtsSetUserTag
HtsUpdateLinkEx
HtsWrite

HtsAddSelectionItem

Add an item to the selection box.

bool HtsAddSelectionItem(, CtlId, text, value, selected, insert)

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.
string text;	Text string to display for the item. Set this field to null to prompt the user for parameters.
string 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()

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(OutputTo, FileName, out data)

int OutputTo;	Output to:	
	HTML_FILE:	Specify the output file name using the FileName argument.
	HTML_BUF:	The output buffer is returned via the 'data' string parameters.
string FileName;	Output file name. The file name may be prefixed by a subdirectory path. Set to null when the 'OutputTo' argument is HTML_BUF.	
string data;	When saving to a buffer, this string receives the output html text.	

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(MenuId)

int MenuId;	The menu id for the command to execute.

Description: The following pre-defined 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(CtlId)

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(new, id, method, action, ShowDialog)

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
string 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 HtsEditRule (width, thickness, align, repaint)

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.

Htn HtsGetActiveWnd()

Description: This function is useful for determining the current active frame 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

▼

HtsGetFontSize

Get font size at the current cursor location.

int HtsGetFontSize(out IsRelative)

bool 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(FormId, FieldId, out name, out IntData, out TextData)

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.
string name;	(OUTPUT) The name of the field.
int IntData;	(OUTPUT) The pointer to receive the integer data.
string TextData;	(OUTPUT) The pointer to receive the text data.

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 parameters 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:

```
string action,method,name,TextData;
int i,TotalFields,FieldType,IntData;
TotalFields=HtsGetFormInfo(0, out action,out method);
  // Retrieve information about a form id 0.
for (i=0;i<TotalFields;i++) {
  FieldType=HtsGetFieldData(0,i,out name,out IntData,
                            out TextData);
}
```

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:

```
string action,method;
int TotalFields;
TotalFields=HtsGetFormInfo(0, out action, out method);
  // Retrieve information about a form id 0.
```

See Also: `Form Event`

See Also:
HtsGetFormInfo

HtsGetFormInfo

Get the information about a form.

int HtsGetFormInfo(FormId, out action, out method)

int FormId;	The form id for which to retrieve the information.
string action;	(output) This fields receives the 'action' text for the form.
string 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:

```
string action,method;
int TotalFields;
TotalFields=HtsGetFormInfo(0, out action, out method);
  // Retrieve information about a form id 0.
```

See Also: Form Event

See Also:
HtsGetFieldData

HtsGetFrameName

Retrieve the current frame name.

bool HtsGetFrameName(out name)

string 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(out href, out name)

bool HtsGetLinkDataEx(out href, out name, out target)

string href;	The 'href' parameter.

string name;	The 'name' parameter.
string target;	The the frame target for the link.

Return Value: This function returns a True value when successful.

See Also:
HtsUpdateLinkEx

▼

HtsGetParaAttrib

Get current paragraph attributes.

bool HtsGetParaAttrib(out HdrLevel, out ListType, out ListLevel, out ParaAuxId, out MiscType)

int HdrLevel;	This parameter returns the header level. A header level of 0 indicates a non-header paragraph.	
int ListType;	This parameter 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.
int ListLevel;	This parameter returns the list level for the list type of paragraph. A list level of 0 indicates a non-list type of paragraph.	
int 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.	
int 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(type, id, start, out int1, out int2, out str1, out str2)

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.
int int1;	The location to return the first integer data associated with this tag.
int int2;	The location to return the second integer data associated with this tag.
string str1;	The location to return the first string data associated with this tag.
string 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,
string string1,string2;
HtsGetTagData(TAG_USEMAP,PictId,0,out int1,out int2,
  out string1,out 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(out target)

string target;	The current target frame name.

Return Value: This function returns a True value when successful.

See Also:

HtsSetTarget

HtsGetTopWin

Get the Htn object associated with the top window in the current frame set.

Htn HtsGetTopWin()

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 Htn object for the top window.

Return Value: This function returns the Htn object for 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(TagId, out TagText)

int TagId;	Tag id for the user tag to retrieve
string 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

HtsInsertButtonField

Insert a button or check-box field into form.

bool HtsInsertButtonField(name, type, checked, value, repaint)

string 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.
string 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.

See Also:
HtsInsertTextField
HtsInsertSelectField

HtsInsertLink

Insert a hyperlink at the current location.

bool HtsInsertLink(LinkText, LinkPict, LinkHref, LinkName, IsMap, repaint)

string LinkText;	Link text (null terminated) string for a text type link.
string 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.*
string LinkHref;	Href string (null terminated) for the link. This parameter should contain the url for the link
string 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(file, align, IsMap, IsInput, repaint)

string 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. 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 (repaint)

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(name, MultiSelect, NumLines, repaint)

string 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.

See Also:
HtsAddSelectionItem
HtsDelSelectionItem
HtsInsertTextField

HtsInsertTextField

Insert a text field into form.

bool HtsInsertTextField(name, type, NumCols, MaxCols, NumRows, InitText, repaint)

string 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.	
string 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.

See Also:
HtsInsertButtonField

HtsInternetGet

Download a file from internet.

bool HtsInternetGet(url, OutFile, GetFromCache)

string url;	The url of the internet file
string 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()

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)

string 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)

string 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

HtsLparam2String

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

bool HtsLparam2String(lParam, string)

long lParam;	Source parameter in the long form.
string string;	Destination parameter

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
```

HtsMenuEnable

Return the enable/disable status of a menu item.

bool HtsMenuEnable(MenuId)

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(MenuId)

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()

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(repaint)

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(tag)

string 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(InputType, FileName, data, TagName)

int InputType;	Input type:	
	HTML_FILE:	Specify the input file name using the FileName argument.
	HTML_BUF:	Specify the input buffer using the data argument.
string FileName;	Input file name. The file name may be prefixed by a subdirectory path. Set to null when the 'InputType' argument is HTML_BUF.	
string data;	The string containing HTML text. Set to null when the 'InputType' argument is HTML_FILE.	
string 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. 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 fires the Picture event 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
HtsRead(HTML_FILE, "myfile.htm",null,null);
// read an HTML disk file and position at a tag called
```

```
   'begin here'.
HtsRead(HTML_FILE, "myfile.htm",null, "begin here");
// read a memory buffer containing HTML text
string data=ValidHtmlData
HtsRead(HTML_BUF,null,data,null);
```

HtsReformat

Save the document to a temporary buffer and reload.

bool HtsReformat()

Description: This function is useful for re-displaying 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(OutputTo, FileName, out data)

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.
string FileName;	Output file name. The file name may be prefixed by a subdirectory path. Set to null when the 'OutputTo' argument is HTML_BUF.	
string data;	When saving to a buffer, this parameter receives the html data output.	

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(htn)

Htn htn;	The Htn object to set as active control in frame set.

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.

static bool HtsSetBkColor(BkColor)

Color BkColor;	Background color

Description: This function is used to set a non-default background color for a control. This static method must be called before creating an Htn object.

Return Value: This function always returns True.

See Also:
HtsSetNewBkColor

HtsSetBkPict

Set the background picture file.

bool HtsSetBkPict(PictFile)

string 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(target)

string target;	New default target.

Return Value: This function returns a True value when successful.

See Also:
HtsSetFrameName

HtsSetDocTitle

Set the document title.

bool HtsSetTitle(title)

string 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)

string 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.

int HtsSetFlags(SetReset, flags)

bool SetReset;	True to set the given flags, False to reset them.
int flags;	Flags to set or reset. Currently the following flags are available. These flag constants are defined in the hc class within the Subsystems.HT namespace.

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 <div> tag for HTML output.
HFLAG_NO_SPAN_TAG:	Do not output the <span> 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.

Example:

htn.HtsSetFlags(true,hc.HFLAG_NO_STYLE)

HtsSetFlags2

Set the operating flags.

int HtsSetFlags2(SetReset, flags)

bool SetReset;	True to set the given flags, False to reset them.
int flags;	Flags to set or reset. Currently the following flags are available. These flag constants are defined in the hc class within the Subsystems.HT namespace.

HFLAG2_DOCTYPE:	Generate DOCTYPE information in the document header.

Return Value: This function returns the current flag values.

Example:

htn.HtsSetFlags2(false,hc.HFLAG2_DOCTYPE)

HtsSetFont

Set the new font information for an HTML style element.

bool HtsSetFont(font, typeface, PointSize, style, TextColor, TextBkColor, repaint)

int font;	Font id for an HTML style element. Refer to the 'HtsGetFont' method for a list of the font ids.
string typeface;	The font typeface. This buffer should be able to hold upto 32 characters.
int PointSize;	The new point size for the font.
int style;	The new style bits for the font: BOLD: Bold ITALIC: Italic ULINE: Underline STRIKE Strikethrough
Color TextColor;	The new text foreground color.
Color 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:

```
string typeface;
int PointSize;
int style;
Color TextColor, TextBkColor;
HtsGetFont(FONT_H2, typeface, out PointSize, out style,
           out TextColor, out TextBkColor);
HtsSetFont(FONT_H2, "Arial", PointSize, style, Color.Red,
           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
HtsSetLinkInfo

HtsSetFontDlg

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

bool HtsSetFontDlg(font)

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(-1);
```

See Also:
HtsSetLinkInfo

HtsSetFontSize

Set the font size.

bool HtsSetFontSize(size, relative, repaint)

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
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(set, repaint)

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(id)

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(name)

string name;	New frame name.

Return Value: This function returns a True value when successful.

See Also:
HtsGetFrameName

HtsSetHeader

Apply the header property.

bool HtsSetHeader(level, repaint)

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(style, out TextColor)

int style;	The new style bits for the font:	
	BOLD:	Bold
	ITALIC:	Italic
	ULINE:	Underline
	STRIKE	Strikethrough
Color 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:

```
int style;
Color TextColor;
HtsGetLinkInfo(&style, out TextColor);
HtsSetLinkInfo(style, Color.Red);
// The above example modifies the color for the link text.
   The style is left unchanged.
```

See Also:
HtsSetLinkInfoDlg
HtsSetFont

HtsSetLinkInfoDlg

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

bool HtsSetLinkInfoDlg()

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();
```

See Also:
HtsSetLinkInfo
HtsSetFontDlg

HtsSetListEx

Apply the paragraph list property.

bool HtsSetList(type, level, DtDd, repaint)

bool HtsSetListEx(type, level, DtDd, NbrSymbol, repaint)

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 the 'tc' class within the Subsystems.TE name-space.

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

HtsSetMiscParaType

Set Blockquote, Address, or Preformatted text attributes.

bool HtsSetMiscParaType(type, repaint)

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(color, set)

Color 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(color)

Color 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(SetSpace, repaint)

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(PictId, PictType, PictFile)

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 Picture event (chapter: Events) for a list of valid picture types.
string 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 Picture event 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 Picture event. The Picture event is followed by the PictId event, 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(PictId, PICT_BMP, "new.bmp");
```

See Also:
HtsSetBkPict
HtsSetNewTextColor

HtsSetPictFileBase

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

static bool HtsSetPictFileBase(FileBase)

string 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()

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(width, repaint)

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(target)

string 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("");
```

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(TagId, TagText)

int TagId;	Tag id for the user tag to set
string 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

▼

HtsUpdateLinkEx

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

bool HtsUpdateLink(href, name, repaint)

bool HtsUpdateLinkEx(href, name, target, repaint)

string href;	The new 'href' parameter.
string name;	The new 'name' parameter.
string 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(OutputTo, FileName, out data, WriteSelection)

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.
string FileName;	Output file name. The file name may be prefixed by a subdirectory path. Set to null when the 'OutputTo' argument is HTML_BUF.	
string data;	When saving to a buffer, this parameter receives the html data output	
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

Events

The product offers the following events:

DocBase:

The dll fires this event when it encounters the 'base' HTML marker.

void DocBase(string FileBase)

FileBase	The base URL string.

Return Value: None.

BkPict:

The dll fires this event when it encounters the 'body' tag with background picture specification. This event follows immediately before the Picture event which is sent to resolve the picture file name for the background picture.

void BkPict()

Parameters	None.

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

Form:

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

void FireForm(int form)

form:	The form id of the form being submitted.

Application Response: Within the event handler for this event, 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: None.

ChildFrameFile:

The dll may fire this event before reading a document into a frame of a frame-set document.

void ChildFrameFile(ref string name)

name	The file URL.

Application Response: Your application will resolve the URL into a 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 assigns the file path name to the 'name' parameter. If this event is not handled by your application, then the dll will attempt to read the file specified by the original URL.

Return Value: None.

ChildFrameName:

The dll may fire this event before reading a document into a frame of a frame-set document.

void ChildFrameName(string name)

name	The frame name.

Comment: This event is fired before firing the ChildFrameFile event and it denotes the frame into which the next frame file is being read.

Return Value: None.

Link:

The dll fires this event (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.

void FireLink(string url)

url	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.
	The tag name is delimited from the URL string using '#' character. Example:
	`//localhost/c:\mydir\myfile.htm#mytag`
	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:
	`//localhost/c:\mydir\myfile.htm?10,15`
	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: None.

LinkInfo:

This event is similar to the Link event, except that this event is fired in the 'edit' mode. This event is fired for information only.

void LinkInfo(string url)

InternalLink:

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

void InternalLink(string name)

LinkUrl:

The dll fires this event 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 Link event when the user actually activates a link.

void LinkUrl(string url)

url	The current anchor URL string.

Return Value: None.

PictId:

The dll fires this event when reading a new document into the editor window. This event is followed immediately after the Picture event. It informs your application about the picture id of the preceding picture.

void PictId(int pict)

pict	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: None.

Picture:

The dll may fire this event 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.

int Picture(ref string PictFile)

PictFile	The image URL.

Application Response: Your application will resolve the picutre 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 assigns the local picture path name to the PictFile parameter. 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 PictFile was set to

```
//localhost/c:\mydir\mypict.bmp
your application will set PictFile to c:\mydir\mypict.bmp:
PictFile="c:\mydir\mypict.bmp";
and return PICT_BMP.
```

Return Value: Picture type as described above.

SavePicture:

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

void SavePicture(ref string FileName)

FileName	The image URL. The modified picture URL can also be passed using this parameter.

Return Value: None.

FileTItle:

The editor fires this event when it encounters the 'title' control element in the HTML document.

void FileTitle(string title)

title	The title string.

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

```
   this.Text=title; // 'this' pertains to the current form
```

Return Value: None.

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 methods 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.
- 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.

Interaction Between HTN and TERN DLLs

The HTN dll and the TERN dll work together to make an HTML viewer control.

The TERN 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 HTN dll.

The HTN dll performs the following tasks:

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

On initialization the HTN dll modifies the operating parameters for the TERN dll. The initialization of the TERN dll consists of the following tasks.

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 HTN dll also sets up handlers to TERN events to receive the hyperlink messages.

.

▲

Recompile the DLL

Using Make file:

The product includes a make file called make-mc which can be used to recompile the dll. This make file recompiles the product using the command line compiler. Therefore the environment variable must be set properly to access the .NET c# compiler from the command line.

Using Visual Studio:

The product includes the htn.csproj project file which can be loaded into Visual Studio to recompile this product.

Building a new project:

Please follow these steps to build a new Visual Studio project to recompile the htn dll:

Create a new project.

Project Type: Visual c# Project, Templates: Empty Project

Project Name: htn

Now right-click on 'Htn' at the top of the Solution Explorer window,

Select 'Add Existing Items'. Select and add all hts*.cs files.

Add following references:

System.dll

System.Windows.Forms.dll

System.Drawing.dll

System.data.dll

System.xml.dll

mscorlib.dll

Right click on Tern to select the 'Properties' option. Now change these properties:

General->ObjectType to 'Class Library'

Output File: htn24.dll

Wanring level: 3

Base Address: 0x26840000

Default Namespace:

The Default Namespace must be left blank to properly access the program resources such as icons and bitmap.