



# Software License Agreement

TE Edit control

for **HTML/Javascript** and **Java**

Vesion 31

2011-2024

ALL RIGHTS RESERVED BY

SUB SYSTEMS, INC.

3200 Maysilee Street

Austin, TX 78728

**512-733-2525**

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 'larger' than the editor; the target application is not a standalone word-processor; the target application is not a programmer's utility 'like' a text editor; the target application is not a programmer's product to convert to or from the RTF, HTML, PDF and DOCX formats; and the source code (or part) of the editor is not distributed in any form.

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](mailto: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.



## General Overview

The TER editor routine allows a developer to incorporate text editing features into a desktop or a web application. This product is designed to be simple to use. The TER editor offers the following features:

**Character Formatting and Text Color:** The editor allows for multiple fonts and point sizes within a document. The character styles include **bold**, **underline**, *italic*, superscript, subscript, and strikethrough. The text can be painted in multiple colors.

**Paragraph Formatting Features:** Left indentation, right indentation, hanging indentation, centering, justification and double spacing.

**Tab Support:** Left, right, center, and decimal tab positions.

**Imbedded Picture:** The picture can be imported from a disk file or from the clipboard. The editor supports the bitmap, device independent bitmap and metafile picture formats.

**Word Wrapping:** The word wrapping can be enabled or disabled.

**Support for Multiple Section, Multiple Column documents.** The editor also offers the Page Mode feature to edit side-by-side multiple column text.

**Block Highlighting:** The text can be highlighted using character highlighting or line highlighting functions.

**Cut/Paste:** The cut/paste to clipboard is performed using these formats:

Text Format

Rich Text Format

**Printing and Mail/Merge:** The editor can print the selected text or the entire document to

the selected printer. An API function allows your application to print a text buffer without invoking the text window. This process can also replace the field names with data strings.

**Application Programming Interface:** The APIs allow you to insert or retrieve text and format attributes anywhere in the text window. You can also interface to the editor window using message communication.

In most instances, only a single API function is needed to invoke the editor.

**Input and Output Source:** The editor can accept data in a buffer or from a disk file. Likewise, the output can be obtained in a buffer or disk file.

**Input and Output text format:** The editor supports the following file formats:

Text format

Rich Text Format

**File and buffer size:** The editor supports unlimited size text files using Windows' virtual memory capability.

**Text and Picture Frames:** The editor in the Page Mode allows for frames which can be moved and positioned anywhere on a page.

**Other Features:**

Search/replace

Page break and automatic repagination

Page header/footer

Column break

Table support

Optional tool bar

Optional ruler

Optional menu interface

Optional scroll bars

Optional status ribbon

Control over editor window style

Protected and hidden text

Print preview with zoom

Hyperlink link support

Page numbering

Paragraph Spacing

Picture Alignment

Embed other controls

Tracking changes

Commenting



## Getting Started

Unzip teje.zip file into a separate folder, for example c:\demo:

```
Extracting files from .ZIP: j:\upload\eval\teje.zip
Inflating: demo.rtf
Inflating: demo.zip
Inflating: demo_web_tej.zip
Inflating: help.chm
Inflating: help.pdf
Inflating: make-mc
Inflating: readme.txt
Inflating: tej19.jar
Extracting: web_edit.zip

C:\demo>
```

Now use the jar program to unzip the tej26.jar in the demo folder:

```
jar -xvf tej26.jar
```

This would create a new subdirectory called tej where the Tej class files are extracted:

```
Directory of C:\demo
05/24/2012  03:11 PM    <DIR>      .
05/24/2012  03:11 PM    <DIR>      ..
05/24/2012  02:20 PM    <DIR>      com
05/24/2012  03:10 PM           29,856 demo.rtf
05/24/2012  03:10 PM           41,684 demo.zip
05/24/2012  03:10 PM           11,181 demo_web_tej.zip
05/24/2012  03:10 PM          413,641 help.chm
05/24/2012  03:10 PM         1,855,511 help.pdf
05/24/2012  03:10 PM             835 make-mc
05/24/2012  02:59 PM    <DIR>      META-INF
05/24/2012  02:20 PM    <DIR>      netscape
05/24/2012  03:10 PM           1,263 readme.txt
05/24/2012  03:11 PM    <DIR>      tej
05/24/2012  03:10 PM         2,853,187 tej19.jar
05/24/2012  03:10 PM           1,222 web_edit.zip
               9 File(s)          5,208,380 bytes
               6 Dir(s)    157,725,184,000 bytes free

C:\demo>
```

'tej' is also the name of the package for this product.

Now, unzip the demo.zip file in the same folder (c:\demo) to extract the demo class and source files.

Finally, use the following command to run the demo:

```
java -cp c:\demo; demo
```

### In This Chapter

[Differences between the HTML/Javascript and Java Versions](#)

[License Key](#)

[Creating an Editor Window](#)





## Differences between the HTML/Javascript and Java Versions

This manual describes two separate products, TE Edit Control for HTML/Javascript and TE Edit Control for Java. Many of the APIs are common among these two products (hence the common manual), here are some of the salient differences.

	TE Edit Control for HTML/Javascript	TE Edit Control for Java
Control File Name	terh31.js	tej31.jar and tej31.java
Additional files needed to run the editor	ter_tlb.gif, wait.gif, ter_hlp.pdf	None
Technical help file name	help.chm, help.pdf	help.chm, help.pdf
Demo file name	terh31.htm	demo.java and dmo.html
ASP.NET wrapped control name	WebTeh.dll	WebTer.dll
Server OS	Any	Any
Coding language	Javascript	Java
Hosting Environment	Browser	Java desktop or browser
Browser element to host the editor	canvas	applet
Browser support	almost all major browsers	IE11 or older only
Additional steps needed to host within a browser	None.	The jar must be signed
Ease of use within the browser.	Very simple since the javascript is fully supported within a HTML 5 compliant browser.	Need to tackle security issues which vary from java version to version
Constant prefix	tc. Example, tc.ID_PASTE, tc.TP_CUR_LINE	Object reference, example, tej.TP_CUR_LINE
Drag/drop within the editor	Full support	Full support
Drag/drop from external application	Browser does not allow for the security reason	Full support
Clipboard support within	Full support	Full support

the editor.

Clipboard access from external applications.	Browser allows only the plain text format.	Full support
File system	The browser allows the editor to import the documents and picture files from the local disk using a browser file dialog. Similarly, the text can be written to the file system as a downloaded file.  The browser does not allow non-user mediated access to the local file system for security reasons.	Conventional file system
Windows Metafile and Windows Bitmap format support	Limited on Windows OS and not available on other OS.	Unlimited on Windows OS, available on other OS
Linked picture within the RTF file.	Linked picture with file-system references are not supported since the browser limits access to the file system.	Full linked picture support



## License Key

### TE Edit Control for HTML/Javascript

Please apply the product key as described in your license delivery e-mail.

### TE Edit Control for Java

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

```
Tej.TerSetLicenseKey("www-xxxxx-yyyy-zzzz")
```

Replace the 'www-xxxxx-yyyy-zzzz' by your license key.



## Creating an Editor Window

This chapter describes how to create the editor session for the HTML/Javascript and Java versions of the Edit Control.

### In This Chapter

[TE Edit Control for HTML/Javascript](#)

[TE Edit Control for Java](#)



## TE Edit Control for HTML/Javascript

This topic describes the method of *directly* inserting the editor control into your HTML page using the 'canvas' element. The product also contains an ASP.NET wrapper dll called WebTeh.dll. If you are using ASP.NET, please refer to the [ASP.NET Interface](#) topic for embedding the editor control into your ASP.NET page.

The distribution file contains a number of files. The editor code is contained in the terh31.js file. Please copy the following files to your project folder, such as: c:\inetpub\wwwroot\myapp

terh31.js	The main editor javascript file.
ter_tlb.gif	This file contains the toolbar icons
wait.gif	This image is displayed as a wait icon
ter_hlp.pdf	Editor help file for the user. This file is displayed when the user clicks on the 'help' icon.

The distribution zip file also contains a demo file called terh.htm.

Here are the basic steps:

1. Add a script function called **ControlCreated** which shall be called after the control is created. Here you will add statements to customize the control for your need. You can also add a statement to catch the event handler function used to catch the events should be physically placed before the 'ControlCreated' function so browser loads these in proper order.
2. Insert a statement to load the main script called **terh31.js**.
3. Insert the **'canvas'** tag.
4. Call the **TerInit** function to create the editor in the 'canvas' element.

5. Call the **ControlCreated** function (step#1).

Here is an example:

**NOTE: ALL EDITOR CONSTANT, SUCH AS ID\_PASTE, MUST BE PREFIXED WITH A tc. PREFIX**

```
<!-- Event handler functions -->
<script>
    function PreprocessEvent(obj,ActionType,ActionId) {
        console.log("Preprocess, object id: ",obj.cvs.id,"type",ActionType
            "id: ",ActionId);
    }

    function Hypertext(obj,HText,HCode) {
        if (HCode.indexOf("http://")==0) window.open(HCode);
        else if (HCode.indexOf("www.")==0) window.open("http://"+HCode);
        else alert("Hypertext: "+HText+"\nCode/url: "+HCode);
    }
</script>

<script>
    var tej=null;
    function ControlCreated(te) {
        tej=te; // save in a global variable for easy access

        tej.TerSetEvent("Preprocess",PreprocessEvent); // an example of
                                                    //catching the preprocess e
        tej.TerSetEvent("Hypertext",Hypertext);

    }
</script>

<!-- Load the main editor javascript file -->
<!-- This should be done before the canvas tag -->
<script src="terh31.js"></script>

<div style="margin-left: 50px;">
    <table><tr>
    <td>
        <div>
```

```
<canvas id="ter1" width="1000" tabindex=0 height="800"
    style="border: 1px solid;"
    data-word-wrap = "true"
    data-page-mode = "true"
    data-print-view-mode = "true"
    data-fitted-view = "false"
    data-show-menu = "true"
    data-show-status-bar = "true"
    data-show-ruler = "true"
    data-show-vert-ruler = "true"
    data-show-tool-bar = "true"
    data-border-margin = "true"
    data-rtf-output = "true"
    data-read-only-mode = "false"
    data-vert-scroll-bar = "true"
    data-horz-scroll-bar = "true"

>
    Sorry, this browser does not support HTML 5.
</canvas>
</div>
</td>
</tr></table>
```

```
</div>
```

```
<!-- Create the editor session within the canvas element created above.
    This should be done after the loading the script (above) and
    the canvas element (above);
-->
```

```
<script>
    var tel=TerInit('ter1');    // Create the editor session
    ControlCreated(tel);
</script>
```

Please refer to the included terh.htm for a full example.



## TE Edit Control for Java

The Tej class extends the the Panel class. Here are the steps to add a Tej class object to your application Frame window.

(Please refer to the ASP.NET Interface for a description of the steps needed to insert a Tejw Applet object into your ASP.NET application).

To start, extract the tej31.jar files into one of the folders in the class path:

```
jar -xvf tej31.jar
```

This would create a new subdirectory called tej where the Tej class files are extracted. 'tej' is also the name of the package for this product. The 'import' statement should be used on the top of your application to access the editor classes.

Now follow these steps to insert a Tej control to your application.

```
import tej.*;

public class dmo_te extends Frame implements WindowListener
{

    Tej tej=null;

    JMenuBar MainMenu=null;

    public dmo_te()
    {
        frame=this;

        addWindowListener(this);

        // Set the frame dimensions
        setLocation(x,y);
        setSize(width,height);
        setTitle("Unknown");
    }
}
```

```

// Create tej control

// Only design-time methods should be used here since the
Tej control is not yet created.
// Such code should is added in the OnCreate function below

tej = new Tej();
tej.addTejListener(tej_listener);

tej.setWordWrap(true);      // Word wrap
tej.setPageMode(true);     // page mode
tej.setPrintViewMode(true); // print-view mode
tej.setFittedView(true);   // fitted view mode
tej.setShowRuler(true);    // show the ruler
tej.setShowVertRuler(true); // show vertical ruler
tej.setShowStatusBar(true); // show the status bar
tej.setShowToolBar(true);  // show the toolbar
tej.setVertScrollBar(true); // show vertical toolbar
bar
tej.setHorzScrollBar(true); // show the horizontal scroll

tej.setBorderMargin(true); // create the border margin
tej.setReadOnlyMode(false); // read only

// add following to show the menu
MainMenu=tej.TerBuildMenu();

MainMenu.setBorder(BorderFactory.createLineBorder(Color.black));
add(MainMenu, BorderLayout.NORTH);

// add the Tej control, the control is actually constructed
in the WindowActivated event
add(tej); // add tej control to your Frame

/*****
*****
OnCreate:

```

Here you can add additional code that relies on Tej control handle being already created. The function is called from the

windowActivated method located below.

```
*****  
*****/
```

```
void OnCreate()  
{  
    tej.ReadTerFile(MyFile.rtf); // read the initial file  
into the  
                                control  
    setTitle(MyFile.rtf); // show the file name on the title  
}
```

```
////////////////////////////////////  
// Handle events using the TejAdapter class. Please refer to  
the  
// 'Control Events' topic for the description of all methods  
exported // by this adapter class.
```

```
////////////////////////////////////  
TejListener tej_listener = new TejAdapter() {  
    // provide overrides for the event which you would like to  
use  
    // this event fires when the user clicks on a hyperlink  
and  
    when the Hyperlink cursor is made visible  
  
    public void Hypertext(TejEvent ev) {  
        JOptionPane.showMessageDialog(frame, ev.text, ev.code,  
JOptionPane.PLAIN_MESSAGE);  
    }  
  
    // this event fires when a new file is opened in the  
control  
    public void OpenFile(TejEvent ev) {  
        frame.setTitle(ev.DocName);  
    }  
};
```

```
////////////////////////////////////
```



```

//////////
    // WindowListener methods

////////////////////////////////////
//////////

    public void windowActivated(WindowEvent ex) {
        if (!tej.TerControlCreated()) {
            tej.TerCreateControl(); // construct the Tej control,
if
            // not already constructed

            OnCreate();
        }
    }

    public void windowClosed(WindowEvent ex) {}
    public void windowClosing(WindowEvent ex) {
        if (tej!=null && tej.TerIsModified() &&
!tej.TerGetReadOnly()) {
            int MessageResult;
            if (JOptionPane.YES_OPTION==(MessageResult=
                JOptionPane.showConfirmDialog(this,"File
modified.
                Do you wish to save the Modifications?","",
JOptionPane.YES_NO_CANCEL_OPTION))) {
                if (!tej.SaveTerFile(null)) return;
            }
            else if (MessageResult==JOptionPane.CANCEL_OPTION)
return;
        }

        // close window now
        Window w = ex.getWindow();
        w.setVisible(false);
        w.dispose();
    }

    public void windowDeactivated(WindowEvent ex) {}
    public void windowDeiconified(WindowEvent ex) {}
    public void windowIconified(WindowEvent ex) {}
    public void windowOpened(WindowEvent ex){}

```



## Control Methods

These API functions allow you to open, close and manipulate data in a TE window.

### Note for TE Edit Control for HTML/Javascript:

The example supplied with the control methods in this chapter use the tej. prefix which is applicable only to the Java version of TE Edit Control. *For TE Edit Control for HTML/Javascript, all program constants must be prefixed by a tc. namespace, example: tc.ID\_PASTE, tc.TP\_CUR\_LINE.*

### Note for TE Edit Control for Java:

The 'Tej' class is defined in the *SubSystems.TE* name space.

Before a TE control is created, please set the [license key](#) to unlock the full functionality of the product.

### In This Chapter

[Import and Export](#)

[Text Insert, Append and Delete](#)

[Print and Print-preview](#)

[Character Formatting](#)

[Paragraph Formating](#)

[Section Formatting](#)

[Document](#)

[Text Selection](#)

[Cursor and Text Position](#)

[Table](#)

[Hyperlink](#)

[Mail-merge](#)

[Picture and Embedded Controls](#)

[Page Header/Footer](#)

[Frame and Drawing Objects](#)

[Footnote, Endnote, Bookmark, Tag](#)

[Stylesheet](#)

[Control Flags](#)

[List Numbering](#)

[Toolbar](#)

[Undo](#)

[Input Field](#)

[Page](#)

[Search, Replace, Locate](#)

[Track Changes](#)

[Comment](#)  
[Menu Command](#)  
[Screen Drawing](#)  
[Spell Checking](#)  
[HTML Add-on Interface](#)  
[Miscellaneous](#)



## Import and Export

This chapter includes the data import and export functions.

### In This Chapter

[getData](#)  
[GetTerBuffer](#)  
[ReadTerFile](#)  
[SaveTerFile](#)  
[setData](#)  
[SetTerBuffer](#)  
[TerDocName](#)  
[TerFileToMem](#)  
[TerGetReadOnly](#)  
[TerIsModified](#)  
[TerSetModify](#)  
[TerSetOutputFormat](#)  
[TerSetReadOnly](#)  
[TerQueryExit](#)



### getData

#### Retrieve document data:

String GetData()

**Description:** You can use this function to retrieve the window text and format data in a string. This function can be called any time for a TEJ window. The format of the data within the buffer is controlled by the current output format set by the TerSetOutputFormat function.

**Return Value:** This function returns a String containing the text and format data.

A null return value indicates an error.

This method is functionally the same as the GetTerBuffer method.



## GetTerBuffer

### Retrieve document data:

String GetTerBuffer()

**Description:** You can use this function to retrieve the window text and format data in a string. This function can be called any time for a TEJ window. The format of the data within the buffer is controlled by the current output format set by the TerSetOutputFormat function.

**Return Value:** This function returns a String containing the text and format data.

A null return value indicates an error.

This method is functionally the same as the getTerBufferData method.

### See Also

[SetTerBuffer](#)

[TerSetOutputFormat](#)



## ReadTerFile

### Load a file into the editor window

boolean ReadTerFile(FileName)

String FileName;                      // Name of the disk file to read.

**Description:** This function instructs the editor to load the specified file for editing. Any existing text in the window is discarded.

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

### See Also:

[SaveTerFile](#)

[GetTerBuffer](#)

[SetTerBuffer](#)



## SaveTerFile

### Save the editor text to disk

boolean SaveTerFile(FileName)

String FileName; // Name of the disk file to save text.

**Description:** This function instructs the editor to save the document data to the specified disk file. The format of the output data can be specified by first calling the TerSetOutputFormat function.

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

#### See Also:

[ReadTerFile](#)

[GetTerBuffer](#)

[SetTerBuffer](#)

[TerSetOutputFormat](#)



## setData

### Set document text:

void setData(buffer)

String buffer; // The String containing the new text and format data.

**Description:** You can use this function to set new data in an existing TEJ window. *The existing text in the window is discarded.* The data in the buffer can be provided in one of these formats:

Text Format

Rich Text Format

**Return Value:** None



## SetTerBuffer

### Set document text:

boolean SetTerBuffer(buffer, title)





## TerFileToMem

**A utility function to read a file into a character array.**

```
char[] TerFileToMem(FileName)
```

**Description:** This function simply reads a file into a character. Please note that this function does not load the file into the editor's buffer. To load a file into the editor, please use the function: ReadTerFile.

**Return Value:** This function returns the character array containing the file. A null value indicates an error condition.



## TerGetReadOnly

**Retrieve the current Read Only status.**

```
boolean TerGetReadOnly()
```

**Return Value:** The function returns true if read-only mode is turned on.

**See Also:**

[TerSetReadOnly](#)



## TerIsModified

**Check if the editor data needs saving.**

```
boolean TerIsModified()
```

**Return Value:** This function returns true if the text is modified and is not yet saved.

**See Also:**

[TerSetModify](#)



## TerSetModify

**Set or reset the modification flag.**

```
boolean TerSetModify(modify)
```

boolean modify; // true to set the modification flag, false to reset it

**Description:** This flag is used to set or reset the modification flag. The modification flag is used to prompt the user to save the data before exiting the editor.

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

**See Also:**

[TerIsModified](#)



## TerSetOutputFormat

**Set the format of the output data.**

boolean TerSetOutputFormat(format)

boolean format; // The output format can be set to one of the following:

SAVE_TEXT:	Save in the text format
SAVE_TEXT_LINES:	Save in the text format with line breaks.
SAVE_RTF:	Save in the Rich Text Format
SAVE_HTML:	Save in the HTML format. This option is available only when HTML Add-on is installed. The <a href="#">HTML Add-on license key</a> must also be set for this flag to be fully effective.  (reserved for future use)
SAVE_UTEXT:	Save in the Unicode text format. (not available in the 16 bit product)

**Description:** This function is used to change the data format of the document when saved. The original output format is specified by the calling application when the editor window is created.

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



## TerSetReadOnly



### Set Read Only status.

boolean TerSetReadOnly( ReadOnly)

boolean ReadOnly; // (true/false) New status of the Read Only flag

**Description:** This function is used to set or reset the Read Only status.

**Return Value:** The function returns the previous value of the Read Only status.

#### See Also:

[TerGetReadOnly](#)



## TerQueryExit

**Check if it is OK to close a TE window.**

boolean TerQueryExit()

**Return Value:** This function returns true if it is OK to close a TE window. Otherwise it returns a false value.



## Text Insert, Append and Delete

### In This Chapter

[InsertRtfBuf](#)

[InsertTerText](#)

[SetTerLine](#)

[TerAddAutoCompWord](#)

[TerAppendText](#)

[TerAppendTextEx](#)

[TerDeleteBlock](#)

[TerGetLine](#)

[TerGetLineInfo](#)

[TerGetLineParam](#)

[TerGetLineWidth](#)

[TerGetRtfSel](#)

[TerInsertLine](#)

[TerInsertRtfFile](#)

[TerInsertText](#)

[TerSetLine](#)



## InsertRtfBuf

### Insert a text buffer in the RTF format at the specified cursor location.

boolean InsertRtfBuf(buffer, line, column, repaint)

String buffer; // The String containing RTF data

int line; // line location (base 0) where the text will be inserted. You can also set the 'line' parameter to -1 to insert the rtf document at the current cursor position. Or, set it to -2 to insert the rtf file at the end of the current document.

int column; // column location (base 0) where the text will be inserted. This parameter is ignored when the 'line' parameter is set to -1 or -2.

boolean repaint; // true to refresh the window after this operation

**Description:** This function is used to insert a buffer containing the text (in RTF format) into the specified TEJ window. The text is inserted at the specified line and column position.

To specify the location in terms of the line and column numbers, specify the line number in the 'line' argument and column number in the 'column' argument. To specify the absolute location, set the 'column' argument to -1, and set the 'line' argument to the absolute text location. To insert the text at the current cursor location, set the 'line' and 'col' arguments to -1. To append the text at the end of the document, set the 'line' to -2.

This function is available in the 'word wrap' mode only.

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

#### See Also:

[InsertTerText](#)  
[GetTerBuffer](#)  
[TerInsertRtfFile](#)



## InsertTerText

### Insert a text buffer in the ASCII format at the current cursor location.

boolean InsertTerText( buffer, repaint)

String buffer; // String containing ASCII text data

boolean repaint; // true to refresh the window after this operation

**Description:** This function is used to insert a String containing the text (in ASCII format) into the specified TEJ window. The text is inserted at the current cursor position.

The text lines within the buffer must be delimited using CR/LF pair.

The buffer must be terminated using a null character.

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

#### See Also:

[InsertRtfBuf](#)

[TerAppendText](#)

[SetTerBuffer](#)

[SetTerCursorPos](#)



## SetTerLine

This function has been replaced by the [TerSetLine](#) function.



## TerAddAutoCompWord

**Set an auto-completion word/phrase pair.**

```
boolean TerAddAutoCompWord(ACWord, ACPhrase)
```

```
String ACWord;           // An auto-completion key word.
```

```
String ACPhrase;        // The auto-completion phrase for the key word.
```

**Description:** This function adds the key word and the expansion phrase to the auto-completion list. When the user type a key word, the control automatically replaces it with the corresponding phrase.

**Result:** This function returns True when successful.



## TerAppendText

**Append specified text at the end of the buffer.**

```
boolean TerAppendText( text, FontId, ParalId, repaint)
```

```
String text;             // String containing text to be appended.
```

```
int FontId;             // font id to use for the new text. Use -1 for the default
```

value.

int Parald; // paragraph id to use for the new text. Use -1 for the default value.

boolean repaint; //Repaint the window after this operation

**Description:** This function adds the specified text at the end of the buffer. The current cursor position does not change after the insertion.

This is a very efficient function which can be used to rapidly build a document. This function does not attempt to wrap the text as they are being added. Your application should call the TerRewrap or TerRepaginate functions after making a series of calls to this function.

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

**See Also:**

[TerAppendTextEx](#)

[TerInsertText](#)

[TerCreateFont](#)

[TerCreateParald](#)



## TerAppendTextEx

**Append specified text at the end of the buffer (enhanced version).**

boolean TerAppendTextEx( text, FontId, Parald, CellId, reserved, repaint)

String text; // String containing the text to be appended.

int FontId; // font id to use for the new text. Use -1 for the default value.

int Parald; // paragraph id to use for the new text. Use -1 for the default value.

int CellId; // Cell id to use for this new text. Use -1 for the default value. When creating a table structure using this function, you must specify the current cell id for this parameter.

int ParaFID; // Paragraph frame id. Use -1 for the default value.

boolean repaint; //Repaint the window after this operation

**Description:** Please refer to the TerAppendText function for more information.

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

**See Also:**[TerAppendText](#)[TerInsertText](#)[TerInsertLine](#)[TerCreateFont](#)[TerCreateParald](#)

## TerDeleteBlock

**Delete a highlighted text block.**

```
boolean TerDeleteBlock( repaint)
```

```
boolean repaint;           // repaint the screen after this operation.
```

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



## TerGetLine

**Retrieve a text line.**

```
int TerGetLine( LineNo)
```

```
int LineNo;                // line number (0 to TotalLines - 1) to retrieve.
```

**Description:** This routine is used to retrieve the text and the font ids for a specified line. The 'font' variable receives an array of font ids. You can get further information about each font id by calling the 'GetFontInfo' function.

**Return Value:** This function returns the length of the retrieved line. This function return -1 if an error is encountered or when the 'LineNo' exceeds the total lines in the windows. The line text and the associated font array can be retrieved using the [TerGetOutIntArray](#) method.

**C# Example:**

```
String text;  
int[] font;  
int len;  
  
' Get the text and font ids  
if (TerGetLine(LineNo)>0) {
```

```
text=tej.TerGetOutStrArray("text"); // get the line text.  
  
font=tej.TerGetOutIntArray("FontArray"); // get the  
           // integer array from the previous call to  
           // TerGetLine method  
}
```

There is one font-id associated with each character on the line. Therefore the string length of the line-text would be the same as the length of the font id array.

#### See Also:

[TerSetLine](#)  
[TerGetLineInfo](#)  
[TerGetParaInfo](#)  
[GetFontInfo](#)



## TerGetLineInfo

### Get the current line attributes.

boolean TerGetLineInfo(LineNo);

int LineNo;                               Line number to retrieve the information. Set to -1 to use the current line number.

**The following information can be retrieved by using the [TerGetOutInt](#) method after a successful return from this method.**

int ParaId;                               The paragraph id. Retrieve as following:  
if (TerGetLineInfo(LineNo)) {  
    ParaId = TerGetOutInt("ParaId");  
}

int CellId;                               The table cell id. Retrieve as following:  
if (TerGetLineInfo(LineNo)) {  
    CellId = TerGetOutInt("CellId");  
}

int ParaFID;                              The paragraph frame id. Retrieve as following:  
if (TerGetLineInfo(LineNo)) {

```
ParaFID = TerGetOutInt("ParaFID");  
}
```

int x; The x position (in twips) of the line. Retrieve as following:

```
if (TerGetLineInfo(LineNo)) {  
    x = TerGetOutInt("x");  
}
```

int y; The y position (in twips) of the line relative to the top of the page. Retrieve as following:

```
if (TerGetLineInfo(LineNo)) {  
    y = TerGetOutInt("y");  
}
```

int height; The height (in twips) of the line.

Retrieve as following:

```
if (TerGetLineInfo(LineNo)) {  
    height = TerGetOutInt("height");  
}
```

int lflags; Line flags as defined by the LFLAG\_ constants:

```
tej.LFLAG_PARA : paragraph or cell  
break  
line  
tej.LFLAG_BREAK : any other break line  
tej.LFLAG_PARA_FIRST : first para line  
tej.LFLAG_CONTROL : line has embedded  
control  
tej.LFLAG_FNOTE : contains a footnote  
tej.LFLAG_SOFT_COL: soft column break  
tej.LFLAG_NBSpace : line has non-break  
space characters  
tej.LFLAG_LINE : line break  
tej.LFLAG_HTML_RULE: is html horizontal  
rule  
tej.LFLAG_BOX_TOP : draw top para  
border  
before this line  
tej.LFLAG_BOX_BOT : draw bot para  
border  
before this line  
tej.LFLAG_SECT : section break line
```





```
line
tej.LFLAG_AUTONUMLGL: has an autonumlgl
field in the line
```

Retrieve as following:

```
if (TerGetLineInfo(LineNo)) {
    lflags = TerGetOutInt("lflags");
}
```

int InfoFlags;

Miscellaneous flags as defined by the INFO\_ constants:

```
tej.INFO_TAB           = has TAB
tej.INFO_SECT          = has section break
tej.INFO_PAGE          = has page break
tej.INFO_COL           = has column break
tej.INFO_CELL          = has cell break
tej.INFO_ROW           = has row break
tej.INFO_TABLE         = has table
tej.INFO_JUST          = justified text
tej.INFO_FRAME         = displaced by a
frame
tej.INFO_SPACE_LINE    = is frame space
tej.INFO_PAGE_NUMBER  = has page number
tej.INFO_FRM_SPC_BEF  = has space before
frame
tej.INFO_DYN_FIELD    = has dynamic field
```

Retrieve as following:

```
if (TerGetLineInfo(LineNo)) {
    InfoFlags =
    TerGetOutInt("InfoFlags");
}
```

**Description:** The x, y, and the height values are available in the Page Mode only.

**Return Value:** This function returns a true value when successful. Otherwise it returns false. The TerGetOutInt parameter can be used to retrieve various bit of information after a successful return from this method:

```
if (TerGetLineInfo(LineNo)) {
    CellId = TerGetOutInt("CellId");
    ParaId = TerGetOutInt("ParaId");
}
```

```
}
```

**See Also:**

[TerGetLineWidth](#)



## TerGetLineParam

**Get a parameter associated with a text line number.**

```
int TerGetLineParam(LineNo, type)
```

```
int LineNo;           // Line number (0 to TotalLines -1) to retrieve parameters.
```

```
int type;             // The parameter to retrieve:
```

LP_LINE_NUM	Current line number.
LP_LINE_LEN	Return the line length.
LP_CELL_ID	Cell id associated with the line. A non-zero value indicates that the line is included in a table.
LP_PARA_FRAME_ID	Paragraph frame id associated with the line. A non-zero value indicates that the line is included in a positionable object.
LP_LIST_ID	List id associated with the line. A non-zero value indicates that the line is included in a list.
LP_LIST_OR_ID	List override id associated with the line. A non-zero value indicates that the line is included in a list.
LP_LIST_LEVEL	List nesting level.
LP_LIST_FONT	Font id associated with the bullet text on this line.

**Return Value:** The function returns the value for the requested parameter. It returns LP\_ERROR to indicate an error condition.



## TerGetLineWidth

**Get the line width.**

```
int TerGetLineWidth(LineNo)
```

```
int LineNo; // line number (0 to TotalLines - 1) to get width.
```

**Return Value:** This function returns the line width in twips. A return value of -1 indicates an error condition.

### See Also

[TerGetLineInfo](#)



## TerGetRtfSel

**Retrieve the selected text in the RTF format.**

```
String TerGetRtfSel()
```

**Return Value:** This function returns the String containing the selected text in the RTF format.

A null value of the handle indicates an error.

### See Also:

[TerGetTextSel](#)

[GetTerBuffer](#)

[SetTerBuffer](#)

[ReadTerFile](#)

[SaveTerFile](#)



## TerInsertLine

**Insert a line of text before the current line.**

```
boolean TerInsertLine( text, FontId, ParaId, CellId, reserved, repaint)
```

```
String text; //the text to be appended.
```

```
int FontId; // font id to use for the new text. Use -1 for the default value.
```

```
int ParaId; // paragraph id to use for the new text. Use -1 for the
```

default value.

int CellId; // Cell id to use for this new text. Use -1 for the default value. When creating a table structure using this function, you must specify the current cell id for this parameter.

int ParaFID; // Paragraph frame id. Use -1 for the default value.

boolean repaint; //Repaint the window after this operation

**Description:** This function provides a very fast method of inserting a line of text before the current line. After the insertion the cursor is moved to the first character of the next line. In a batch of calls to the TerInsertLine function, the 'repaint' argument should be set to true only for the last call to this function.

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

#### See Also:

[TerAppendText](#)

[TerInsertText](#)

[TerCreateFont](#)

[TerCreateParald](#)



## TerInsertRtfFile

**Insert an RTF file at the specified cursor location.**

boolean TerInsertRtfFile( FileName, line, column, repaint)

String FileName; // The name of the RTF file.

int line; // line location (base 0) where the text will be inserted. You can also set the 'line' parameter to -1 to insert the rtf document at the current cursor position. Or, set it to -2 to insert the rtf file at the end of the current document.

int column; // column location (base 0) where the text will be inserted. This parameter is ignored when the 'line' parameter is set to -1 or -2.

boolean repaint; // true to refresh the window after this operation

**Description:** Please refer to the InsertRtfBuf function for further description of the 'line' and 'column' arguments.

This function is available in the 'word wrap' mode only.

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

**See Also:**

[InsertRtfBuf](#)  
[SetTerBuffer](#)



## TerInsertText

**Insert text at the cursor position.**

```
boolean TerInsertText( text, FontId, Parald, repaint)
```

```
String text;                // the text to be inserted.  
  
int FontId;                 // font id to use for the new text. Use -1 for the default  
                           // value.  
  
int Parald;                 // paragraph id to use for the new text. Use -1 for the  
                           // default value.  
  
boolean repaint;           //Repaint the window after this operation
```

**Description:** This function inserts the specified text at the current cursor position. After the operation the cursor is positioned after the inserted text.

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

**See Also:**

[TerAppendText](#)  
[InsertTerText](#)  
[TerCreateFont](#)  
[TerCreateParald](#)



## TerSetLine

**Set the text and font ids for a line**

```
boolean TerSetLine( LineNo, text, font)
```

```
int LineNo;                 // The text line number (0 to TotalLines - 1) to set  
  
String text;                // the text String to apply.  
  
int[] font;                 // Contain the font id array to apply. You can set this  
                           // parameter to null if the font ids are not available.
```

**Description:** This function is used to set new text for a line. The length of the 'font' array

must be identical to the number of characters in the text string. The font array must contain valid font ids (0 to TotalFonts - 1). You can get information about an editor font id by using the GetFontInfo function.

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

**See Also:**

[TerGetLine](#)

[GetFontInfo](#)



## Print and Print-preview

### In This Chapter

[TerIsPrinting](#)

[TerMergePrint](#)

[TerOverridePageSize](#)

[TerPrint](#)

[TerPrintPreview](#)

[TerSelectPrint](#)

[TerSetDefPrinter](#)

[TerSetPreview](#)

[TerSetPrinter](#)

[TerSetPrintPreview](#)

[TerSpoolBegin](#)

[TerSpoolEnd](#)



## TerIsPrinting

**Check if the editor is printing a document.**

boolean TerIsPrinting()

**Return Value:** This function returns true if the editor is printing or previewing a document.

**See Also:**

[TerPrint](#)



## TerMergePrint

**Print a document without invoking the editing session.**

static boolean TerMergePrint(prt)

TejPrint prt; // Print request parameter structure.

**Description:** This function is used to print a buffer or a file to a specified printer or window device context and at a specified location on the page. If requested, this function can replace the field names with field data. The print parameter class (TejPrint) is used to specify the printing parameters.

```
class TejPrint {
    char        InputType;
    String      file;
    String      buffer;
    Graphics    gr;
    boolean     IsPrinter
    Rectangle   rect;
    boolean     FullPage;
    int         StartPos;
    boolean     OnePage;
    int         NextPos;
    String      MergeFields;
    String      MergeData;
    boolean
PrintHiddenText;
    boolean
PrintMarginArea;
    Form        parnt;
    int         NextY;
}
```

#### **Member Variables:**

InputType:	This flag specifies the input type. If you wish to specify a disk file name, set the 'InputType' to 'F'. Conversely, if you wish to pass the text in a buffer, set this field to 'B'.
file:	If the 'InputType' is 'F', specify the input file path name in this field.
buffer:	If the 'InputType' is 'B', specify the document data using this variable..
gr:	The graphics object of the printer or the window to print the text. Set to null to print to the default printer.
IsPrinter:	Set to true if the graphics object specified by the 'gr' member variabe corresponds to a printer.
rect:	This field contains the rectangle co-ordnates to print to. The rectangle co-ordinates are specified in the millimeter

(mm) units. For a higher precision, you can specify the rectangle coordinates in twips units by specifying the negative values for the left, right, top and bottom variables in the rect structure.

FullPage	Print full page. This argument is used only when printing to the default printer (gr = null); The 'rect' variable is not used when FullPage is set to true.
StartPos:	The positive value for this field specifies the character position to begin the printing. The negative value specifies the page number to begin printing. Set to 0 to begin the printing from the first page.
OnePage:	Set this variable to true to print one page only. When this variable is set to false, the editor prints the entire document by spooling each page.
NextPos:	(OUTPUT) The editor returns the character position of the next page to be printed. When the entire document is printed, the editor sets this field to 0.
MergeFields:	This field specifies the variable to a list of mail merge field names. Each field name must be separated by a ' ' character. The list must be terminated by a null character. If you do not wish to merge field data, set this field to null.
MergeData:	This field specifies the variable to a list of mail merge data strings. Each data String must be separated by a ' ' character. The number of data elements in the 'MergeData' array MUST be the same as the number of elements in the 'MergeFields' array. The list must be terminated by a null character. If you do not wish to merge field data, set this field to null.
PrintHiddenText:	Set this field to true to print any hidden text.
PrintMarginArea:	Print margin and header/footers.
parent:	Parent window form.
NextY:	(OUTPUT) The field returns the Y pixel position on the device context after printing.

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

#### See Also:

[TerPrint](#)  
[TerMergeFields](#)  
[TerSetDefPrinter](#)  
[TerSetInitTypeface](#)





## TerOverridePageSize

**Override the document page size.**

*static* boolean TerOverridePageSize(width, height)

int width; // The page width in twips unit (for the portrait orientation)

int height; // The page height in twips unit (for the portrait orientation)

**Description:** This function is used to specify a page size which may not be supported by the selected printer. The editor formats the document to the given page size, but it does **not** enforce the page size to the printer. This function is called before an editor window is opened. Set the width and height to 0 to reset to the regular page size for the subsequent editor windows.

**Return Value:** This function returns the true when successful.

**See Also:**

[TerSetPaper](#)



## TerPrint

**Print the current document.**

boolean TerPrint(dialog)

boolean TerPrintEx(dialog, FirstPage, LastPage)

boolean TerPrint2(dialog, FirstPage, LastPage, Copies, Colate)

boolean dialog; // Set to true to show a dialog box to the user.

int FirstPage; // First page (zero based) to print. Set to -1 to disable page range printing. Set to -2 to print the current page. Set to -3 to print the selected text if text is selected, or the entire document if text is not selected. This parameter is ignored if the 'dialog' parameter is set true.

int LastPage; // Last page (zero based) to print. This parameter is ignored if the 'FirstPage' parameter is set to -1,-2, -3, or the 'dialog' parameter is set to true.

int Copies; // Number of copies to print Set to 0 to assume the 'copies' value from the printer setup information.

```
int Colate; // When printing multiple copies, set this parameter to true to print all pages of the first set, before printing the next copy.
```

**Description:** This function can be used to print the current document. When the 'dialog' parameter is true, the editor displays a dialog box. The dialog box allows the user to print the entire document or the selected text.

The text is printed to the currently selected printer using the current settings for margins. The editor creates a new printer device context for printing.

**Return Value:** The editor returns a true value when the print job is successfully completed.

#### See Also:

[TerMergePrint](#)

[TerIsPrinting](#)



## TerPrintPreview

**Preview the specified page in the current document.**

```
boolean TerPrintPreview( gr, rect, page, scale)
```

```
Graphics gr; // The graphics object on which to display the preview output.
```

```
Rectangle rect; // The rectangle on the device context where the preview output is to be displayed. The rectangle must be specified in the device units.
```

```
int page; // Page number to preview (0 to TotalPages-1). Specify a -1 value to preview the current page.
```

```
boolean scale; // Set to true if you wish this API to perform scaling to fit the page within the rectangle. Set to false if your application performs the required scaling.
```

**Description:** This function is used to draw the image of a page at the specified location on the specified device context.

**Return Value:** The editor returns a true value when successful

#### See Also:

[TerPrint](#)

[TerSetPrintPreview](#)

[TerSetPreview](#)



## TerSelectPrint

**Print the selected text.**

boolean TerSelectPrint()

**Description:** This function extracts the selected text and uses the TerMergePrint function to print it to the current printer. Since this function creates a temporary buffer for the selected text, it is not efficient for huge text selection.

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

### See Also:

[TerPrint](#)

[TerMergePrint](#)



## TerSetDefPrinter

**Set the default printer for the editor.**

*static* boolean TerSetDefPrinter(PrtSvc,PageFmt)

PrintService PrtSvc;                   // Default print service object

PageFormat PageFmt;                   // Default page format object

**Description:** Use this function to override the default printer. The new default printer is effective for the controls created after this function is called. This function can also be called before calling the TerMergePrint function.

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



## TerSetPreview

**Set preview parameters.**

boolean TerSetPreview( NumPages, ZoomPct, ShowToolbar)

int NumPages;                         // number of preview pages to display. This value must be a 1 or 2.

int ZoomPct;                         // Zoom percentage (0 to 200). A value of 0 displays full pages fitted in the window area.

```
boolean ShowToolBar;           // Show the print preview tool bar. This setting takes
                                effect after the current print preview session.
```

**Return Value:** The function returns true when successful.



## TerSetPrinter

### Set new printer selection information.

```
boolean TerSetPrinter( PrtSvc, PageFmt, ModDoc)
```

```
PrintService PrtSvc;           // New PrintService object
```

```
PageFormat PageFmt;           // New PageFormat object.
```

```
boolean ModDoc;                // Set to true to update the current document using the
                                page-size, page-orientation and bin values from the
                                specified PageSettings object. The values are applied to
                                all sections in the document.
```

**Description:** This function is used to set the new printer selection information.

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



## TerSetPrintPreview

### Allocate or deallocate print preview resources.

```
boolean TerSetPrintPreview( begin)
```

```
boolean begin;                 // true to allocate print preview resources, and false to
                                deallocate them.
```

**Description:** This function can be called before and after calling the TerPrintPreview function to increase the print preview performance when doing multi-page print-preview.

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

### Example:

```
tej.TerSetPrintPreview(true); // begin print preview process
for (page=0;page<PageCount;page++){
    tej.TerPrintPreview(gr,rect,page,true); // print each page
```

```
}  
    tej.TerSetPrintPreview(false); // end print preview process
```

**See Also:**

[TerPrintPreview](#)



## TerSpoolBegin

**Begin a multi-document print job.**

*static boolean* TerSpoolBegin(name)

String name; // The name of the print job. The print-job appears with this name in the printer queue.

**Description:** This method together with the TerSpoolEnd method is used to combine multiple calls to the [TerMergePrint](#) method into a single print job. When calling the [TerMergePrint](#) method, the member variable 'gr' within the TejPrint class must be set to null.

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

**Example:**

```
Tej.TerSpoolBegin("MyPrintJob") // begin the print job  
  
Tej.TerMergePrint(...) // print the first document  
Tej.TerMergePrint(...) // print the second document  
Tej.TerMergePrint(...) // print the third document  
  
Tej.TerSpoolEnd() // end the print job
```



## TerSpoolEnd

**End a multi-document print job.**

*static boolean* TerSpoolEnd(name)

String name; // The name of the print job. The print-job appears with

this name in the printer queue.

**Description:** This method together with the TerSpoolBegin method is used to combine multiple calls to the [TerMergePrint](#) method into a single print job. When calling the [TerMergePrint](#) method, the member variable 'gr' within the TejPrint structure must be set to null.

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

**Example:**

```
Tej.TerSpoolBegin("MyPrintJob") // begin the print job

Tej.TerMergePrint(...) // print the first document
Tej.TerMergePrint(...) // print the second document
Tej.TerMergePrint(...) // print the third document

Tej.TerSpoolEnd() // end the print job
```



## Character Formatting

### In This Chapter

- [GetFontInfo](#)
- [SetTerBkColor](#)
- [SetTerCharStyle](#)
- [SetTerColor](#)
- [SetTerDefaultFont](#)
- [SetTerPointSize](#)
- [SetTerFont](#)
- [TerCreateFont](#)
- [TerGetCurFont](#)
- [TerGetEffectiveFont](#)
- [TerGetFontAux1Id](#)
- [TerGetFontLang](#)
- [TerGetFontFieldId](#)
- [TerGetFontParam](#)
- [TerGetFontSpace](#)
- [TerGetFontStyleId](#)
- [TerGetTextColor](#)
- [TerLocateFontId](#)
- [TerLocateStyle](#)
- [TerLocateStyleChar](#)
- [TerRestrictFont](#)
- [TerSelectCharStyle](#)
- [TerSetCharAuxId](#)

[TerSetCharLang](#)  
[TerSetCharScaleX](#)  
[TerSetCharSet](#)  
[TerSetCharSpace](#)  
[TerSetDefLang](#)  
[TerSetDefTextColor](#)  
[TerSetEffectiveFont](#)  
[TerSetFontId](#)  
[TerSetFontSpace](#)  
[TerSetFontStyleId](#)  
[TerSetInitTypeface](#)  
[TerSetNextFontAux1Id](#)  
[TerSetTcField](#)  
[TerSetTextCase](#)  
[TerSetUlineColor](#)  
[TerSetWaveUnderline](#)  
[TerShrinkFontTable](#)



## GetFontInfo

**Retrieve information about an editor font id.**

**boolean GetFontInfo( FontId)**

boolean GetFontInfo2( FontId)

int FontId;

// The editor font id to retrieve. A valid font id would a value between 0 and TotalFonts - 1.

You can also use this function to get the font information for a stylesheet item by specifying the style id as a negative value for this parameter. Also, use the following constant to specify normal or the current style item.

SID\_NORMAL:      Normal style item

SID\_CUR:          Style item being current edited

**The following information can be retrieved by using the [TerGetOutInt](#) and [TerGetOutStr](#) methods upon a successful return from this method.**

String typeface;

// This variable receives the typeface for the specified font id. The typeface String can be up to 32 characters. Retrieve as following:

```
if (GetFontInfo(FontId)) {  
    typeface = TerGetOutStr("typeface");  
}
```

int PointSize;

// This integer variable receives the point size for the

specified font id. This argument is used by the GetFontInfo function only. Retrieve as following:

```
if (GetFontInfo(FontId)) {  
    PointSize =  
    TerGetOutInt("PointSize");  
}
```

int TwipsSize;

// This integer variable receives the size of the specified font in the twips unit. This is useful when a font uses fractional point size (1 point equal 20 twips). This argument is used by the GetFontInfo2 function only. Retrieve as following:

```
if (GetFontInfo2(FontId)) {  
    TwipsSize =  
    TerGetOutInt("TwipsSize");  
}
```

int style;

// This integer variable receives the styles flags for the specified font id. Retrieve as following:

```
if (GetFontInfo(FontId)) {  
    style = TerGetOutInt("style");  
}
```

The style flag consists of the following bits:

BOLD:	Bold
ULINE:	Underline
ULINED:	Double Underline
ITALIC:	Italic
STRIKE:	Strikethrough
DOUBLE_STRIKE	Double line strike
SUPSCR:	Superscript
SUBSCR:	Subscript
HIDDEN:	Hidden Text
PROTECT:	Protected Text
CAPS	All caps



**Description:** Use this function to retrieve information about an editor font id.

**Return Value:** This function returns true if successful. The function returns false if an error is encountered or when the FontId specifies an invalid font id.

#### See Also

[TerGetLine](#)

[GetTerFields](#)

[TerGetFontStyleId](#)



## SetTerBkColor

**Set the background color for the text.**

**boolean SetTerBkColor(color,repaint)**

boolean SetTerBkColorHtml(HtmlColor,repaint)

Color color; // new background color.

String HtmlColor; The color value can be passed as a Color object, or an HTML color string such as red, "#ff0000".

boolean repaint; // true to repaint the screen after this operation.

**Description:** If a text block is selected before this operation, the new background color is applicable for every character in the block. If a block is not highlighted, this function selects the new color for the next character input.

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

#### See Also:

[SetTerColor](#)

[TerGetTextColor](#)



## SetTerCharStyle

**Set or reset the character styles**

boolean SetTerCharStyle( styles, OnOff, repaint)

int styles; // Character style to set or reset:

BOLD:	<b>Bold</b>
ULINE:	<u>Underline</u>
ULINED:	Double Underline
ULINE_DOTTED	Dotted underline
ITALIC:	<i>Italic</i>
STRIKE:	Strikethrough
DOUBLE_STRIKE	Double strikethrough
SUPSCR:	Superscript
SUBSCR	Subscript
HIDDEN:	Hidden Text
PROTECT:	Protected Text
HLINK:	Hyperlink
CAPS:	All caps
SCAPS:	Small Caps

To specify more than one styles, use the 'logical OR' (|) operator.

```
boolean OnOff;           //true to set the specified styles, false to reset the
                        //specified styles.
```

```
boolean repaint;        //true to refresh the window after this operation
```

**Description:** This function is used to set or reset the give character styles. If a text block is highlighted, this operation is applicable to all characters in the block. Otherwise only the current character is affected.

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

**Example:**

```
tej.SetTerCharStyle(tej.BOLD,true,true); // turn-on the BOLD style
```

#### See Also:

[SelectTerText](#)

[TerLocateStyle](#)

[TerSetUlineColor](#)



## SetTerColor

### Set text color

```
boolean SetTerColor( color, repaint)
```

```
Color color;                // new color to apply.  
String HtmlColor;          The color value can be passed as a Color object, or an  
                            HTML color string such as red, "#ff0000".
```

```
boolean repaint;           //true to refresh the window after this operation
```

**Description:** This function is used to apply the new color to the text. If a text block is highlighted, this operation is applicable to all characters in the block. Otherwise the new color is selected for the next character input.

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

### Example:

```
tej.SelectTerText(0,0,0,40,false); // select the first 40 characters of the first line.  
tej.SetTerColor(Color.Green,true); // apply green color to the selected text
```

### See Also:

[SetTerBkColor](#)  
[TerGetTextColor](#)  
[SelectTerText](#)



## SetTerDefaultFont

### Set the default font and pointsize

```
boolean SetTerDefaultFont( typeface, PointSize, style, color, repaint)
```

```
boolean SetTerDefaultFontHtml( typeface, PointSize, style, HtmlColor, repaint)
```

```
String typeface;           // typeface of the default font
```

```
int PointSize;             // Point size for the default font. You can also specify the  
                            font size in twips by specifying a negative value (1 point  
                            equal 20 twips)
```

```
int style;                 // style of the default font. Refer to the SetTerCharStyle  
                            function for a list of style constants. Use 0 for default.
```

```
Color color;               // default text color. Use 0 for default.
```

```
String HtmlColor;           // Color specified in the html String format, such as "red",
                             "#FFFFFF"

boolean repaint;           //true to refresh the window after this operation
```

**Description:** This function is used by an application to change the initial or default font for the editor. Any text that used the previous default font now gets printed with the new font, point size, style and color.

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

#### Example:

```
// use Arial typeface and 12 pointsize for default.
SetTerDefaultFont("Arial",12, Color.red, true);

// use Arial typeface and 12 pointsize
SetTerDefaultFont("Arial",-240,Color.red, true);
```

**See Also:**  
[SetTerFont](#)



## SetTerPointSize

### Set font pointsize for the text

```
boolean SetTerPointSize( pointsize, repaint)
```

```
int pointsize;           // size of the new font in points (72 points = 1 inch). You
                             can specify the font size in twips unit by using a negative
                             value (1 point equal 20 twips).
```

```
boolean repaint;           //true to refresh the window after this operation
```

**Description:** This function is used to apply the new font size to the text. If a text block is highlighted, this operation is applicable to all characters in the block. Otherwise only the current character is affected.

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

#### Example:

```
// select the first 40 characters of the first line.
SelectTerText(0,0,0,40,false);
```

```
// Set the point size of 14 to the selected text
SetTerPointSize(14,true);

// Set the font size to 280 twips (14 points).
SetTerPointSize(-280,true);
```

#### See Also:

[SelectTerText](#)  
[SetTerFont](#)  
[SetTerCharStyle](#)



## SetTerFont

### Set font typeface for the text

boolean SetTerFont( typeface, repaint)

String typeface; // typeface of the new font

boolean repaint; //true to refresh the window after this operation

**Description:** This function is used to apply the new font typeface to the text. If a text block is highlighted, this operation is applicable to all characters in the block. Otherwise only the current character is affected.

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

### Example:

```
// select the first 40 characters of the first line.
SelectTerText(0,0,0,40,false);

// apply Arial font typeface to the selected text
SetTerFont("Arial",true);
```

#### See Also:

[SelectTerText](#)  
[SetTerPointSize](#)  
[SetTerCharStyle](#)  
[SetTerDefaultFont](#)



## TerCreateFont

### Create a font id.

```
int TerCreateFont( ReuseId, shared, typeface, pointsize, style, color, BkColor, FieldId,
AuxId)

int TerCreateFont2( ReuseId, shared, typeface, pointsize, style, color, BkColor, FieldId,
AuxId, CharStyld, ParaStyld, expand)

int TerCreateFont3( ReuseId, shared, typeface, pointsize, style, color, BkColor, FieldId,
AuxId, CharStyld, ParaStyld, expand, CharSet)

int TerCreateFont3( ReuseId, shared, typeface, pointsize, style, HtmlColor, HtmlBkColor,
FieldId, AuxId, CharStyld, ParaStyld, expand, CharSet)

int ReuseId;                // Existing font id to modify with new information. Use -1
                            // to create a new font id.

boolean shared;            // When true, the editor matches the requested
                            // specification against the existing font ids. If a matching
                            // font id is found, it returns that id. Otherwise it creates a
                            // new id. A true value for this field is mutually exclusive
                            // with a zero or positive value for the ReuseId field.

String typeface;          // Typeface of the new font

int PointSize;            // Point size of the new font. You can also specify the
                            // font size in twips unit by using a negative value (1 point
                            // equal 20 twips).

int style;                // Style bits for the new font. Refer to the function
                            // SetTerCharStyle for a list of style ids. Use 0 for the
                            // default value.

Color color;              // Text foreground color. use 0 for default.

String HtmlColor          // Text foreground color in Html string format, example
                            // "red", "#FF0000"

Color BkColor;            // Text background color. use 0xFFFFFFFF for default.

String HtmlBkColor        // Text background color in Html string format, example,
                            // "red" or "#FF0000"

int FieldId;              // Text field id. Use 0 for the default value.

int AuxId;                // An application specified id. The editor does not use this
                            // id intejally. Use 0 for default.

int CharStyld;            // Character style id. Set to 1 to use the default.

int ParaStyld;            // Paragraph style id. Set to 0 to use the default.

                            // Character width expansion in twips unit. Use 0 for
```

```
int exapnd;                default.  
  
int CharSet                // Character set for the font
```

**Description:** This function is used to create a new font id or to modify an exiting id with new font information. To modify an existing id, specify the old font id using the 'Reuseld' argument, otherwise set the 'Reuseld' parameter to -1. When an existing id is modified, this function automatically updates the text which uses this id with new information.

**Return Value:** When successful, this function returns the id of the new font. Otherwise it returns -1.

**See Also:**

[TerCreateParald](#)

[TerAppendText](#)

[TerSetNextFontAux1Id](#)



## TerGetCurFont

**Retrieve the font id (or picture id) at the given location.**

```
int TerGetCurFont( LineNo, ColNo)
```

```
int LineNo;                // line number for the location. Set to -1 to use the  
                           // current text line and column position.
```

```
int ColNo;                 // column number for the location.
```

**Return Value:** This function returns the font id for the character at the given location.

**Note:** For a picture character, the font ids is same as the picture id. So the font id returned by the TerGetCurFont function is actually a picture id if the LineNo/ColNo parameters point to a picture.

**See Also:**

[TerGetEffectiveFont](#)



## TerGetEffectiveFont

**Retrieve the font id effective at the current cursor position.**

```
int TerGetEffectiveFont()
```

**Return Value:** This function returns the effective font id for the next keyboard input at the current cursor position. If your application uses an external toolbar, use this font id to show the current font attributes in your toolbar.

**See Also:**

[TerGetCurFont](#)  
[GetFontInfo](#)



## TerGetFontAux1Id

**Retrieve the Aux1Id id associated with a Font id.**

```
int TerGetFontAux1Id( FontId)
```

```
int FontId; // Font id to inquire
```

**Return Value:** This function returns the Aux1Id associated with a font id..

**See Also:**

[TerSetNextFontAux1Id](#)



## TerGetFontLang

**Retrieve the language id for a font id.**

```
boolean TerGetFontLang( FontId)
```

```
int FontId; // FontId (0 to TotalFonts-1). Set to 0 to specify the  
default font.
```

**Return Value:** This function returns the language id for the font id. The language ids are defined by MS RTF Spec 1.5 or later.

**See Also:**

[TerSetCharLang](#)



## TerGetFontFieldId

**Retrieve the field id associated with the given font id.**

```
int TerGetFontFieldId( FontId)
```

```
int FontId; // FontId (0 to TotalFonts-1). Set to 0 to specify the  
default font.
```

**Return Value:** This function returns the field id associated with a font id. A value of 0 indicates that the font id is not used for a field text.



**See Also:**  
[TerGetFontSpace](#)



## TerGetFontParam

**Retrieve the font parameters.**

```
int TerGetFontParam( FontId, type)
```

```
boolean TerGetFontParam( FontId, type, out color)
```

```
int FontId; // FontId (0 to TotalFonts-1). Set to 0 to specify the default font.
```

You can also use this function to get the font information for a stylesheet item by specifying the style id as a negative value for this parameter. Also, use the following constant to specify normal or the current style item.

```
SID_NORMAL 'Normal' style item.
```

```
SID_CUR Style item being currently edited.
```

```
int type; // One of the following parameter types can be used:
```

```
FONTINFO_CHARSET The character set for the font
```

```
FONTINFO_PICT_WIDTH The picture width in twips assuming that the FontId represents a picture.
```

```
FONTINFO_PICT_HEIGHT The picture height in twips assuming that the FontId represents a picture.
```

```
FONTINFO_ULINE_COLOR The color of the underline bar.
```

```
FONTINFO_AUX_ID The auxiliary id for the font.
```

```
FONTINFO_FLAGS Font flags. The following font flags are available:  
FFLAG_AUTO_SPELL: Font with wave underline.
```

```
FONTINFO_FRAME_TYPE The frame type for a floating picture assuming that the FontId represents a picture..
```

Please refer to the [TerSetPictFrame2](#)

	function for a list of PFRAME_ constants.
FONTINFO_FRAME_ID	The frame id for a floating picture assuming that the FontId represents a picture.
FONTINFO_OFFSET	Return the character offset (twips) from the baseline.
FONTINFO_IS_PICT	Returns 1 if the FontId represents a picture, otherwise returns 0.
FONTINFO_IS_CTL	Returns 1 if the FontId represents a control, otherwise returns 0.
FONTINFO_FIELD	Field id used for the font.
FONTINFO_INS_REV_ID	Reviewer id for inserted text.
FONTINFO_DEL_REV_ID	Reviewer id for deleted text.
FONTINFO_SIZE	Return the font styles. Please refer to the GetFontInfo function for the definition of style constants.
FONTINFO_STYLE	Return the font style constants. Please refer to the GetFontInfo function for the font style constant values.
FONTINFO_TEXT_COLOR	Return the text color
FONTINFO_BK_COLOR	Return the text background color
FONTINFO_WLINE_COLOR	Return the color of the wave line under the text.

Color color;                      The 'out' parameter to retrieve the color parameter. This parameter is valid only for the parameter types which return a color value.

**Return Value:** This first method returns the value of the requested parameter. It returns -1 to indicate an error condition.

The second override method returns True if successful, otherwise it returns a False value.



## TerGetFontSpace

**Retrieve the character spacing for a font id.**

```
boolean TerGetFontSpace( FontId)
```

```
int FontId; // FontId (0 to TotalFonts-1). Set to 0 to specify the default font.
```

**Return Value:** This function returns the font space adjustment for the font id in Twips unit.

### See Also:

[TerSetFontSpace](#)



## TerGetFontStyleId

**Retrieve the character style id associated with a Font id.**

```
int TerGetFontStyleId( FontId)
```

```
int FontId; // Font id to inquire
```

**Return Value:** This function returns the character style id associated with a font id.

### See Also:

[GetFontInfo](#)  
[TerSelectCharStyle](#)  
[TerEditStyle](#)  
[TerSetFontStyleId](#)



## TerGetTextColor

**Get the foreground and background colors for the specified font id.**

```
boolean TerGetTextColor( FontId)
```

```
int FontId; // The editor font id to retrieve color. A valid font id would a value between 0 and TotalFonts - 1.
```

You can also use this function to get the text color for a stylesheet item by specifying the style id as a negative value for this parameter. Also, use the following constant to specify normal or the current style item:

SID\_NORMAL: Normal style item  
SID\_CUR: Style item being current edited

The following information can be retrieved by using the **TerGetOutColor** ([TerGetOutColor, http://www.](#)) and **TerGetOutStr** methods upon a successful return from this method

Color TextColor;	Get the text foreground color. Example: <pre>if (TerGetTextColor(FontId)) {     Color TextColor=TerGetOutColor("TextColor"); }</pre>
String TextColor	Get the text foreground color in Html string format. Example: <pre>if (TerGetTextColor(FontId)){     String TextColor=TerGetOutStr("TextColor"); }</pre>
Color TextBkColor;	Get the text background color. Example: <pre>if (TerGetTextColor(FontId)) {     Color BackColor=TerGetOutColor("TextBackColor"); }</pre>
String TextBkColor;	Get the text background color in Html text format. Example: <pre>if (TerGetTextColor(FontId)) {     String BackColor=TerGetOutStr("TextBackColor"); }</pre>

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

#### See Also:

[SetTerColor](#)  
[SetTerBkColor](#)  
[GetFontInfo](#)  
[GetTerFields](#)



## TerLocateFontId

**Locate a font id in the document.**

boolean TerLocateFontId(FontId,line, col)

int FontId; // Font id to locate.

int line; // Starting line number. Set to -1 to start the search from the current cursor position.

Upon a successful return from this method, you can retrieve the new 'line' value using the [TerGetOutInt](#) method where given the font-id is found:

```
line=tej.TerGetOutInt("Line");
```

int col; // Starting column position. Set the 'line' and 'col' arguments to 0 to begin the search from the beginning of the file.

Upon a successful return from this method, you can retrieve the new 'col' value using the [TerGetOutInt](#) method where given the font-id is found

```
col=tej.TerGetOutInt("Col");
```

**Return Value:** This function returns a true value if the font id is found in the document. The updated line/col value can be retrieved by using the TerGetOutInt method.

**See Also:**

[TerLocateStyle](#)



## TerLocateStyle

**Locate text with the given character style.**

boolean TerLocateStyle(style, ref StartLine, ref StartCol, out StringLen)

WORD style;	// Style bits:
BOLD:	<b>Bold</b>
ULINE:	<b>Underline</b>
ULINED:	Double Underline
ITALIC:	<i>Italic</i>
STRIKE:	Strikethrough
SUPSCR:	Superscript
SUBSCR:	Subscript
HIDDEN:	Hidden Text

PROTECT:	Protected Text
CAPS	All Caps
SCAPS	Small Caps
PICT	Picture

Use the logical OR (|) operator to specify more than one styles. The search is successful when any of the specified styles are located.

int StartLine: (INPUT/OUTPUT) Specifies the line number to start the search. On a successful search, you can retrieve the new 'StartLine' value using the [TerGetOutInt](#) method where a character with the given style is found:

```
line=tej.TerGetOutInt("Line");
```

int StartCol: (INPUT/OUTPUT) Specifies the column number to start the search. On a successful search, you can retrieve the new 'StartCol' value using the [TerGetOutInt](#) method where a character with the given style is found:

```
col=tej.TerGetOutInt("Col");
```

On a successful search, you can retrieve the following additional information using the [TerGetOutInt](#) method:

int StringLen: The length of the located text. The editor matches the text up to the end of the line. Example:

```
length=tej.TerGetOutInt("StringLen");
```

**Description:** Use his function to locate the beginning of the text with the given character styles.

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

#### See Also:

[TerLocateStyleChar](#)

[TerLocateFontId](#)

[SetTerCharStyle](#)



## TerLocateStyleChar

**Locate the character with the given style.**

boolean TerLocateStyleChar(style, present, StartLine, StartCol, forward)

int style; // Style bits. See TerLocateStyle function for the detail.

boolean present;	// true to test for the presence of the given style, or false to test for the absence of the given style.
int StartLine:	(INPUT/OUTPUT) Specifies the line number to start the search. On a successful search, you can retrieve the new 'StartLine' value using the <a href="#">TerGetOutInt</a> method where a character with the given style is found:  StartLine=tej.TerGetOutInt("StartLine");
int StartCol:	(INPUT/OUTPUT) Specifies the column number to start the search. On a successful search, you can retrieve the new 'StartCol' value using the <a href="#">TerGetOutInt</a> method where a character with the given style is found:  StartCol=tej.TerGetOutInt("StartCol");
boolean forward;	// true to scan the text in the forward direction, or false to scan the text in the backward direction.

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

**See Also:**

[TerLocateStyle](#)



## TerRestrictFont

**Restrict the font size and styles.**

boolean TerRestrictFont(MinSize, MaxSize, RestrictStyles, UpdateToolbar)

int MinSize;	// The smallest point-size of the fonts to allow
int MaxSize;	// The largest point-size of the fonts to allow
int RestrictStyles;	// Use this parameter to specify the styles not to be allowed. The styles are specified by using the style bits. Please refer to the <a href="#">SetTerCharStyle</a> method for a list of the valid style bits.
boolean UpdateToolbar;	// TRUE to update the tool-bar immediately to reflect the changes.

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



## TerSelectCharStyle

**Apply a character stylesheet item.**

boolean TerSelectCharStyle( StyleId, repaint)

int StyleId; // Character style id to apply

boolean repaint; // true to refresh the screen after this operation.

**Description:** This function is used to assign the given character style to a highlighted block of text. If a text block is not highlighted, the given style id is used for the next keyboard input.

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

### See Also:

[TerSelectParaStyle](#)

[TerEditStyle](#)

[TerGetFontStyleId](#)



## TerSetCharAuxId

**Set the character auxiliary id.**

boolean TerSetCharAuxId( AuxId, repaint)

int AuxId; // Aux id

boolean repaint; // Repaint the screen after this operation

**Description:** This function is used to set an auxiliary id. The editor does not use this id. If a text block is selected, the new id is applied to all characters in the block. Otherwise, the id is applied to any newly entered characters at the current cursor location.

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



## TerSetCharLang

**Set new language id for the selected text.**

boolean TerSetCharLang( lang, repaint)



```
int lang;                // Language id as defined by MS RTF Spec 1.5.

boolean repaint;        // Repaint the screen after this operation
```

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

**See Also:**

[TerGetFontLang](#)

[TerSetDefLang](#)



## TerSetCharScaleX

**Adjust horizontal character scaling.**

```
boolean TerSetCharScaleX(dialog, ScalePercent, repaint)
```

```
boolean dialog;          // TRUE to show the dialog box to accept the user input

int ScalePercent;       // Horizontal character scaling in percentage. Set to a
                        // value greater than 100 to enlarge the character
                        // horizontally or set to a value less than 100 to shrink the
                        // character horizontally. The default value is 100.

boolean repaint;        // Repaint the screen after this operation
```

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



## TerSetCharSet

**Override the character set for font creation.**

```
boolean TerSetCharSet( NewCharSet)
```

```
int NewCharSet;         // New character set. To reset the override, set the
                        // NewCharSet to DEFAULT_CHARSET or 1.

boolean repaint;        // Repaint the screen after this operation
```

**Description:** Use this function to override the character-set that the editor should use to create new fonts.

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



## TerSetCharSpace

**Adjust the character spacing.**

boolean TerSetCharSpace( dialog, delta, repaint)

boolean dialog; // true to show the dialog box to accept the user input

int delta; // Amount of adjustment in twips. Set to a positive value to expand the character spacing or a negative value to compress the character spacing.

boolean repaint; // Repaint the screen after this operation

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

**See Also:**

[TerSetFontSpace](#)



## TerSetDefLang

**Set default language id for the document.**

boolean TerSetDefLang( lang)

int lang; // Language id as defined by MS RTF Spec 1.5.

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

**See Also:**

[TerGetFontLang](#)

[TerSetCharLang](#)



## TerSetDefTextColor

**Set default text foreground color.**

boolean TerSetDefTextColor( ForeColor, repaint)

boolean TerSetDefTextColor( HtmlForeColor, repaint)

```
Color ForeColor;           // new text foreground color.

String HtmlForeColor;     // specify the new text foreground color in the html
                           // format, such as "red", "#FF0000".

boolean repaint;         // true to repaint the screen after this operation.
```

**Description:** The new text color is effective only for the current session. The new text color is not written out when the document is saved.

**Please use the `GetTerFields/SetTerFields` functions to change the background color of the text window.**

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

## TerSetEffectiveFont

**Override the effective font for editing.**

```
boolean TerSetEffectiveFont(FontId)

int FontId;           // The font id to use for editing.
```

**Description:** Normally, when the user click on a text location, the editor picks a font for text entry at that location. You can override this font automatic selection by using this function. This function can also be used to set a suitable font before inserting the text using the `InsertTerText` function.

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



## TerSetFontId

**Set the object id for the next object to be inserted.**

```
boolean TerSetFontId( int NextId)

int NextId;           // The id to use by the next object to be inserted.
```

**Description:** Normally, the editor assigns a new id when an object (font or picture) is inserted in the text. This function can be used to utilize an existing id for the object. The object at the existing id is released before associating the new object with this id.

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



## TerSetFontSpace

**Adjust the character spacing for a font id.**

boolean TerSetFontSpace( FontId, delta, repaint)

int FontId; // FontId (0 to TotalFonts-1) to modify. Set to 0 to modify the default font.

int delta; // Amount of adjustment in twips. Set to a positive value to expand the character spacing or a negative value to compress the character spacing.

boolean repaint; // Repaint the screen after this operation

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

### See Also:

[TerSetCharSpace](#)

[TerGetFontSpace](#)



## TerSetFontStyleId

**Set the style ids for a font id.**

boolean TerSetFontStyleId( FontId, CharStyleId, ParaStyleId)

int FontId; // FontId (0 to TotalFonts-1) to modify.

int CharStyleId; // New character style id for the font. Set to -1 to leave this value unchanged.

int ParaStyleId; // New paragraph style id for the font. Set to -1 to leave this value unchanged.

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

### See Also:

[TerGetFontStyleId](#)



## TerSetInitTypeface

**Set the initial font typeface.**

boolean TerSetInitTypeface(typeface)

String typeface; // Initial font typeface. The editor uses this font typeface as the default font for the TerMergePrint function. It also uses this typeface if the editor window is created without specifying a font typeface.

**Description:** To be effective, this function must be called before any editor window is created and before calling the TerMergePrint function.

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



## TerSetNextFontAux1Id

**Set the next font Aux1Id to be used by the TerCreateFont function.**

boolean TerSetNextFontAux1Id(Aux1Id)

int Aux1Id; // Aux1Id to set

**Description:** This function sets the value of the Aux1Id font attribute used by the TerCreateFont function. The TerCreateFont function sets this value to 0 after creating a new font id.

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

### See Also:

[TerCreateFont](#)

[TerGetFontAux1Id](#)



## TerSetTcField

**Set the 'tc' field to the selected text.**

boolean TerSetTcField(level, repaint)

int level; // The 'tc' field level. The level value must be between 1 and 9 inclusive.

boolean repaint; // TRUE to repaint the screen after this operation

**Comment:** This function can be used to set the 'tc' field to the text to be included in the table of contents. Please refer to the [TerInsetToc2](#) function for more information.

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



## TerSetTextCase

**Set the case for the selected text.**

```
boolean TerSetTextCase( upper, repaint)
```

```
boolean upper;           // Set to true to turn the text into upper case. Set to false  
                        // to turn the text into lower case
```

```
boolean repaint;        // true to repaint the screen after this operation
```

**Return Value:** This function returns a true value if successful.



## TerSetUlineColor

**Set the color to draw underline.**

```
boolean TerSetUlineColor(color, repaint)
```

```
boolean TerSetUlineColor(HtmlColor, repaint)
```

```
Color color;           // new underline color.
```

```
String HtmlColor;     // new underline color in Html format, such as  
                      // "red", "#FF0000", etc.
```

```
boolean repaint;      // true to repaint the screen after this operation.
```

**Description:** This function does not automatically apply the underline attribute to the text. The SetTerCharStyle function should be used to apply the underline attribute.

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

**See Also**

[SetTerCharStyle](#)



## TerSetWaveUnderline

### **Draw red wavy underline.**

boolean TerSetWaveUnderline( LineNo, StartCol, EndCol, set, repaint)

boolean TerSetWaveUnderline2( color, LineNo, StartCol, EndCol, set, repaint)

boolean TerSetWaveUnderline2( HtmlColor, LineNo, StartCol, EndCol, set, repaint)

Color color; // The color for the wavy underline. The default color is red. This parameter is applicable to the TerSetWaveUnderline2 method only.

String HtmlColor; // The color for the wavy underline in the Html string format. Example, "red", "#FF0000"

int LineNo; // The text line number

int StartCol; // The starting column position for the text

int EndCol; // The ending column position for the text

boolean set; // Set to true to draw the underline. Set to false to erase the underline.

boolean repaint; // Set to true to repaint the screen after this operation

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



## **TerShrinkFontTable**

### **Compress the font table cache.**

boolean TerShrinkFontTable()

Description: This function is useful if you are opening and closing a number of documents under your program's control. This function can be called after opening a new document to release the unused fonts from the font cache.

**Return Value:** This function true when successful.



## **Paragraph Formating**

### **In This Chapter**

[ClearTab](#)

[ClearAllTabs](#)

[ParaHangingIndent](#)  
[ParaIndentTwips](#)  
[ParaLeftIndent](#)  
[ParaRightIndent](#)  
[ParaNormal](#)  
[SetTab](#)  
[SetTerParaFmt](#)  
[TerCreateBulletId](#)  
[TerCreateListBullet](#)  
[TerCreateParald](#)  
[TerCreateTabId](#)  
[TerGetParaInfo](#)  
[TerGetParaParam](#)  
[TerGetTabStop](#)  
[TerSelectParaStyle](#)  
[TerSelectParaText](#)  
[TerSetBullet](#)  
[TerSetBulletEx](#)  
[TerSetBulletId](#)  
[TerSetDefTabWidth](#)  
[TerSetDefTabType](#)  
[TerSetParaAuxId](#)  
[TerSetParaBkColor](#)  
[TerSetParald](#)  
[TerSetParaBorderColor](#)  
[TerSetParaList](#)  
[TerSetParaShading](#)  
[TerSetParaTextFlow](#)  
[TerSetParaIndent](#)  
[TerSetParaSpacing](#)  
[TerSetPflags](#)  
[TerSetTab](#)



## ClearTab

### Clear one tab stop.

boolean ClearTab( TabPos, repaint)

int TabPos; // Tab position (in twips unit) to clear

boolean repaint; //Repaint the window after this operation

**Description:** Use this function to remove a specified tab stop for the selected text.

When a text block is highlighted before calling this function, the selected lines are affected by this function. Otherwise, only the current paragraph is affected by this operation.

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

**See Also**



[ClearAllTabs](#)  
[SetTab](#)



## ClearAllTabs

### Clear all tab stops:

boolean ClearAllTabs(repaint)

boolean repaint;                               //Repaint the window after this operation

**Description:** Use this function to reset all tab stops for the selected text. The tab stops are reset to their default positions.

When a text block is highlighted before calling this function, the selected lines are affected by this function. Otherwise, only the current paragraph is affected by this operation.

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

### See Also

[SetTab](#)  
[ClearTab](#)



## ParaHangingIndent

### Increment or decrement the hanging indents.

boolean ParaHangingIndent( indent, repaint)

boolean indent;                               // true to increment the indentation, false to decrement the indentation

boolean repaint;                               // true to refresh the window after this operation

**Description:** Use this function to increment or decrement the hanging indentation by 1/4 of an inch.

When a text block is highlighted before calling this function, the selected lines are affected by this function. Otherwise, only the current paragraph is affected by this operation.

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

### See Also:

[ParaLeftIndent](#)  
[ParaRightIndent](#)

[ParaNormal](#)



## ParaIndentTwips

**Increment or decrement the paragraph indentation.**

boolean ParaIndentTwips( DeltaLeft, DeltaRight, DeltaFirst, repaint)

```
int DeltaLeft;           // amount (twips) of left indentation to apply

int DeltaRight;         // amount (twips) of right indentation to apply

int DeltaFirst;        // amount (twips) of indentation to apply to the first line
                       // only.

bool repaint;          // true to repaint the screen after this operation.
```

**Description:** This function can be used to increment or decrement the paragraph indentation amount. This function allows you to affect all three indentation parameters simultaneously.

Please note that the left indentation affects the first line of the paragraph as well. To keep the first line from moving, apply the equal amount of negative indentation to the first line.  
Example:

```
ParaIndentTwips(50,0,-50,true);
```

**Return Value:** The function returns true when successful.

### See Also:

[ParaLeftIndent](#)  
[ParaRightIndent](#)  
[ParaHangingIndent](#)  
[TerSetParaIndent](#)



## ParaLeftIndent

**Increment or decrement the left indents.**

boolean ParaLeftIndent( indent, repaint)

```
bool indent;           // true to increment the indentation, false to decrement
                       // the indentation

bool repaint;         // true to refresh the window after this operation
```

**Description:** Use this function to increment or decrement the left indentation by 1/4 of an

inch.

When a text block is highlighted before calling this function, the selected lines are affected by this function. Otherwise, only the current paragraph is affected by this operation.

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

**See Also:**

[ParaHangingIndent](#)

[ParaRightIndent](#)

[ParaNormal](#)



## ParaRightIndent

**Increment or decrement the right indents.**

boolean ParaRightIndent( indent, repaint)

boolean indent; // true to increment the indentation, false to decrement the indentation

boolean repaint; // true to refresh the window after this operation

**Description:** Use this function to increment or decrement the right indentation by 1/4 of an inch.

When a text block is highlighted before calling this function, the selected lines are affected by this function. Otherwise, only the current paragraph is affected by this operation.

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

**See Also:**

[ParaHangingIndent](#)

[ParaNormal](#)



## ParaNormal

**Reset paragraph properties**

boolean ParaNormal( repaint)

boolean repaint; // true to refresh the window after this operation

**Description:** Use this function to reset the paragraph properties.

When a text block is highlighted before calling this function, the selected lines are affected by this function. Otherwise, only the current paragraph is affected by this operation.

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

**See Also:**

[ParaHangingIndent](#)

[ParaRightIndent](#)

[ParaLeftIndent](#)

[SetTerParaFmt](#)



## SetTab

**Set a tab position:**

```
boolean SetTab( TabType, TabPos, repaint)
```

```
int TabType;                // Tab type: TAB_LEFT, TAB_RIGHT, TAB_CENTER,
                             TAB_DECIMAL
```

```
int TabPos;                  // Tab position (in twips unit) to create
```

```
boolean repaint;            //Repaint the window after this operation
```

**Description:** Use this function to create one tab position.

**Note:** This function will be eventually discontinued in favor of the TerSetTab function.

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

**See Also:**

[TerSetTab](#)

[ClearTab](#)

[ClearAllTabs](#)



## SetTerParaFmt

**Set paragraph styles**

```
boolean SetTerParaFmt( styles, OnOff, repaint)
```

WORD styles:

Select paragraph styles:

LEFT:	Left justified paragraph
CENTER:	Centered paragraph
RIGHT_JUSTIFY:	Right justified paragraph
JUSTIFY:	Paragraph justified on both margins
DOUBLE_SPACE:	Double spaced paragraph
PARA_KEEP:	Keep the entire paragraph on the same page.
PARA_KEEP_NEXT:	Keep the last line of the current paragraph and the first line of the next paragraph on the same page.
PARA_BOX_TOP:	Apply top paragraph border
PARA_BOX_BOT:	Apply bottom paragraph border
PARA_BOX_BETWEEN	Draw lines between selected paragraphs.
PARA_BOX_LEFT:	Apply left paragraph border
PARA_BOX_RIGHT:	Apply left paragraph border
PARA_BOX:	Apply paragraph borders to all sides (combination of above four flags)
PARA_BOX_DOUBLE:	Double line paragraph border
PARA_BOX_THICK:	Thick paragraph border

To specify more than one styles, use the 'logical OR' (|) operator.

boolean OnOff; // true to set the styles, false to reset the selected styles.

boolean repaint; // true to refresh the window after this operation

**Description:** Use this function to set or reset the specified styles.

When a text block is highlighted before calling this function, the selected lines are

affected by this function. Otherwise, only the current paragraph is affected by this operation.

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

**Example:**

```
// center the current paragraph
SetTerParaFmt (tej.CENTER, true, true);
```

**See Also:**

[ParaNormal](#)

[TerSetFlags](#)



## TerCreateBulletId

### Create a paragraph bullet/numbering id.

int TerCreateBulletId( IsBullet, start, level, type)

int TerCreateBulletId2( IsBullet, start, level, type, TextBef, TextAft)

int TerCreateBulletId3( IsBullet, start, level, type, TextBef, TextAft, flags)

boolean IsBullet; // true to set the paragraph bullet or false to set paragraph numbering.

int start; // The starting number when setting paragraph numbering. Set to 1 for default.

int level; // The level number when setting paragraph numbering. Set to 0 for default.

int type; // The parameter indicates the symbol used for bullets or the letters used for paragraph numbering. Please refer to the TerSetBulletEx function for the constant symbols used for this parameter.

String TextBef; // Text before the paragraph number (limited to 10 bytes). Set to null for default.

String TextAft; // Text After the paragraph number (limited to one byte). Set to null for default.

int flags; // Set this argument to BLTFLAG\_HIDDEN to create a continuing list item. Set to 0 to create regular list item..

**Return Value:** This function returns the bullet id.

**See Also:**

[TerSetBulletEx](#)

[TerSetBulletId](#)

[TerCreateListBullet](#)



## TerCreateListBullet

**Create a bullet id using the list mechanism.**

**int TerCreateListBullet( ListOr, level)**

int ListOr; // The list-override id to create the bullet.

int level; // The level number to create the bullet. A simple list allows only one list level (level 0). A nested list allows up to 9 levels (0 to 8).

**Return Value:** This function returns a non-zero bullet id when successful. A value of 0 indicates an error condition.

**See Also:**

[TerCreateParald](#)

[TerCreateBulletId](#)



## TerCreateParald

**Create a paragraph id.**

int TerCreateParald( Reuseld, shared, LeftIndent, RightIndent, FirstIndent, TabId, StyleId, AuxId, Shading, pflags, SpaceBefore, SpaceAfter, SpaceBetween, flags);

int TerCreateParaldEx( Reuseld, shared, LeftIndent, RightIndent, FirstIndent, TabId, StyleId, AuxId, Shading, pflags, SpaceBefore, SpaceAfter, SpaceBetween, flags, BltId, BkColor);

int Reuseld; // Existing paragraph id to modify with new information. Use -1 to create a new paragraph id.

boolean shared; // When true, the editor matches the requested specification against the existing paragraph ids. If a matching paragraph id is found, it returns that id. Otherwise it creates a new id. A true value for this field is mutually exclusive with a zero or positive value for the Reuseld field.

int LeftIndent; // Left indentation (specified in twips). Use 0 for default.

int RighthIndent;	// Right indentation (specified in twips). Use 0 for default.
int FirstIndent;	// Indentation for the first line (specified in twips). Use 0 for default.
int TabId;	// Tab id. Use 0 for default.
int StyleId;	// Paragraph Style id. Use 0 for default. When a non-zero style id is specified, other parameters values to the function must be what is indicated by this style id.
int AuxId;	// An application specified id. The editor does not use this id intejjally. Use 0 for default.
int shading;	// Shading amount Specify a value from 0 (no shading) to 10000 (darkest shading).
int pflags;	// Additional paragraph flags reserved for future use. Use 0 for default.
int SpaceBefore;	// Space before the paragraph (specified in twips). Use 0 for default.
int SpaceAfter;	// Space after the paragraph (specified in twips). Use 0 for default.
int SpaceBetween;	// Space between the paragraph lines (specified in twips). Use 0 for default.
int flags;	// Paragraph attribute flags. Please refer to the 'SetTerParaFmt' function for a list of paragraph attribute ids. Use 0 for default.

**When creating a paragraph id for use inside a page header, the 'flags' parameter must be ORed with the PAGE\_HDR constant. Similarly, when creating a paragraph id for use inside a page footer, the 'flags' parameter must be ORed with the PAGE\_FTR constant. A paragraph id for use in the regular text must not have either of these constants.**

int BtId;	// The bullet id. When a non-zero bullet id is specified, the BULLET flag must also be specified in the 'flags' parameter. Set to 0 for default.
Color BkColor:	Paragraph background color. Set to Hex FFFFFFFF (white) for default.  Then BkColor parameter can also be passed in the Html string format, such as "red", "#FF0000",blue, etc.

**Description:** This function is used to create a new paragraph id or to modify an exiting id with new paragraph information. To modify an existing id, specify the old paragraph id



using the 'Reuseld' argument, otherwise set the 'Reuseld' parameter to -1. When an existing id is modified, this function automatically updates the text which uses this id with new information.

**Return Value:** When successful, this function returns the id of the new paragraph. Otherwise it returns -1.

#### See Also:

[TerCreateFont](#)  
[TerAppendText](#)  
[TerGetParaInfo](#)  
[TerSetParald](#)  
[TerCreateTabId](#)  
[TerCreateBulletId](#)  
[TerCreateListBullet](#)



## TerCreateTabId

Create a tab id.

**int TerCreateTabId( TabInfo)**

```
TejTab; // This structure is used to pass the tab stop information
        // for the new tab id. The TejTab class includes these
        // variables:

        int count; // number of tab stop (max 20)

        int pos[20]; // tab position for each tab stop in twips. The tab
        // positions must be specified in the ascending order

        int type[20]; // The tab type for each tab stop:

        TAB_LEFT: Left tab

        TAB_RIGHT: Right tab

        TAB_CENTER: Center tab

        TAB_DECIMAL: Decimal tab

;        BYTE flags[20] // The tab flags for each tab stop:

        TAB_NONE Tab with no leaders (default)

        TAB_DOT: Tab with dot leaders

        TAB_HYPH: Tab with hyphen leaders
```

TAB\_ULINE: Tab with underline leader

**Description:** The tab id created by this function can be used in the TerCreateParald function.

**Return Value:** This function returns a non-zero tab id if successful, otherwise it returns -1.

**See Also:**

[TerCreateParald](#)



## TerGetParalInfo

**Get paragraph attributes.**

boolean TerGetParalInfo( LineNo)

boolean TerGetParalInfo3( LineNo, IsStyleItem)

int LineNo; // Line number to retrieve the paragraph information. To specify a para id instead of a line number, specify a negative value.

boolean IsStyleItem; This parameter is applicable to the TerGetParalInfo3 function only. It allows you to retrieve the paragraph information for a style item. When this flag is set to true, the LineNo parameter should be used to pass a style id to retrieve its information. You can also set the LineNo to SID\_CUR to get the paragraph information about the style being currently edited.

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt, TerGetOutColor, or TerGetOutStirng methods:**

int LeftIndent; // Left indentation (in twips). Example:  
LeftIndent=tej.TerGetOutInt("LeftIndent")

int RighthIndent; // Right indentation (in twips). Example:  
RightIndent=tej.TerGetOutInt("RightIndent")

int FirstIndent; // Indentation for the first line (in twips). Example:  
FirstIndent=tej.TerGetOutInt("FirstIndent")

int TabId; // Tab id. Example:  
TabId=tej.TerGetOutInt("TabId")

int StyleId; // Paragraph Style id. Example:  
StyId=tej.TerGetOutInt("StyId")

int AuxId; // An application specified id. Example:

```

        AuxId=tej.TerGetOutInt("AuxId")

int shading;           // Shading amount: a value from 0 (no shading) to 10000 (darkest
                        // shading). Example:
                        shading=tej.TerGetOutInt("Shading")

int pflags;           // Additional paragraph flags:

                        PFLAG_WIDOW:           Widow/orphan control

                        PFLAG_PAGE_BREAK:     Page break before the paragraph

                        Example:
                        pflags=tej.TerGetOutInt("pflags")

int SpaceBefore;     // Space before the paragraph (in twips). Example:
                        SpaceBefore=tej.TerGetOutInt("SpaceBefore")

int SpaceAfter;      // Space after the paragraph (in twips). Example:
                        SpaceAfter=tej.TerGetOutInt("SpaceAfter")

int SpaceBetween;    // Space between the paragraph lines (in twips). Example:
                        SpaceBetween=tej.TerGetOutInt("SpaceBetween")

int flags;           // Paragraph attribute flags. Please refer to the 'SetTerParaFmt' function
                        // for a list of paragraph attribute ids. Example:
                        flags=tej.TerGetOutInt("Flags")

int Aux1Id;          // Another application specified id. Example:
                        Aux1Id=tej.TerGetOutInt("Aux1Id")

Color BkColor;       // Paragraph background color. Example:
                        Color BkColor=tej.TerGetOutInt("BkColor")
                        or,
                        String HtmlColor = tej.TerGetOutStr("BkColor");

int LineSpacing;     // Extra line spacing in percentage. Example:
                        LineSpacing=tej.TerGetOutInt("LineSpacing")

```

**Return Value:** This function returns a true value when successful. Otherwise it returns false.

#### See Also:

[TerCreateParald](#)

[TerGetParaParam](#)

## TerGetParaParam

Retrieve additional paragraph properties.

```
int TerGerParaParam(LineNo, IsStyleItem, type)
```

```
boolean TerGetParaParamClr(LineNo, IsStyleItem, type)
```

```
int LineNo;           // Line number to retrieve the paragraph information. To specify a  
                      // para id instead of a line number, specify a negative value.
```

```
boolean IsStyleItem; This parameter allows you to retrieve the paragraph information for  
                      // a style item. When this flag is set to TRUE, the LineNo parameter  
                      // should be used to pass a style id to retrieve its information. You can  
                      // also set the LineNo to SID_CUR to get the paragraph information  
                      // about the style being currently edited.
```

```
int type             // Parameter type to retrieve.
```

```
Color color          // Use by the override function to return the color related parameters  
                      // using the TerGetOutColor or TerGetOutStr methods.
```

```
Color color=tej.TerGetOutColor("Color");
```

or

```
String color = tej.TerGetOutStr("Color");
```

```
//return the color in the string format.
```

PARAINFO_TEXT_FLOW	Paragraph text flow. Please refer to the <a href="#">TerSetParaTextFlow</a> function for the list of text flow constants.
--------------------	---

PARAINFO_BORDER_COLOR	Paragraph border color.
-----------------------	-------------------------

PARAINFO_BK_COLOR	Paragraph background color.
-------------------	-----------------------------

**Return Value:** This function returns the value of the requested parameter. It returns PARAINFO\_ERROR to indicate an error condition.

**See Also**

[TerGetParaInfo](#)



## TerGetTabStop

**Return the parameters for a tab stop and the number of tab stop for a line.**

```
int TerGetTabStop( LineNo, TabNo)
```

int TerGetTabStop2( type, LineNo, TabNo)

```
int type; // This argument specifies the meaning of the 'LineNo'
          // argument:

          PID_LINE:      The 'LineNo' parameter specifies the
                          text line number.

          PID_PARA:      The 'LineNo' parameter specifies the
                          paragraph id.

          PID_TAB:       The 'LineNo' parameter specifies the
                          tab id.

int LineNo; // Line number (or paragraph or tab id) to get the tab
            // parameters for. Set to -1 to get the tab stop parameters
            // for the current line.

int TabNo; // Tab number to inquire. To simply get the tab count for
           // the line, set the TabNo parameter to -1.
```

**Upon a successful return from this method, the [TerGetOutInt](#) method can be used to retrieve the temporarily saved values for these variables:**

```
int pPos; // The variable to receive the tab position (in twips).
          // Example:
          int pos=tej.TerGetOutInt("TabPos");

int pType; // The variable to receive the tab type. Example:
          int type=tej.TerGetOutInt("TabType");
          Here are the tab-type constants:

TAB_LEFT:      Left aligned tab

TAB_RIGHT:     Right aligned tab

TAB_CENTER:    Center aligned tab

TAB_DECIMAL:   Decimal aligned tab

int pFlag; // The variable to receive the tab flag. Example:
          int flags=tej.TerGetOutInt("TabFlags");
          Here are the tab-flags constants:

TAB_DOT:       Dot leader

TAB_HYPH:      Hyphen leader
```

TAB\_ULINE: Underline leader

**Return Value:** This function returns the number of tab stops for the line.

**See Also:**

[TerPosTable](#)



## TerSelectParaStyle

**Apply a paragraph stylesheet item.**

boolean TerSelectParaStyle( StyleId, repaint)

int StyleId; // Paragraph style id to apply

boolean repaint; // true to refresh the screen after this operation.

**Description:** This function is used to assign the given paragraph style to the current paragraph. If more than one paragraph is highlighted, then all highlighted paragraphs are assigned the specified paragraph style id.

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

**See Also:**

[TerSelectCharStyle](#)

[TerEditStyle](#)

[TerGetFontStyleId](#)

## TerSelectParaText

**Select entire text in the current paragraph.**

boolean TerSelectParaText(repaint)

boolean repaint; // TRUE to refresh the screen after this operation.

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



## TerSetBullet

**Set the paragraph bullet property.**

boolean TerSetBullet( set, repaint)

```
boolean set;                // true to set the paragraph bullet or false to remove it.

boolean repaint;           // Repaint the screen after this operation
```

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



## TerSetBulletEx

### Set the paragraph bullet/numbering property.

```
boolean TerSetBulletEx( set, IsBullet, start, level, type, repaint)
boolean TerSetBullet2( set, IsBullet, start, level, type, TextBef, TextAft, repaint)
boolean TerSetBullet3( set, IsBullet, start, level, type, TextBef, TextAft, repaint, flags)

boolean set;                // true to set the paragraph bullet/numbering or false to
                           // remove it. The 'start', 'level', and 'type' parameters are
                           // ignored when the 'set' parameter is false.

boolean IsBullet;          // true to set the paragraph bullet or false to set
                           // paragraph numbering.

int start;                 // The starting number when setting paragraph
                           // numbering. Set to 1 for default.

int level;                 // The level number when setting paragraph numbering.
                           // Set to 0 for default.

int type;                  // The parameter indicates the symbol used for bullets or
                           // the letters used for paragraph numbering.
```

When the 'IsBullet' parameter is set to true, use one of the following constants for the 'type' parameter:

```
BLT_ROUND:                Round bullet
BLT_DIAMOND:              Diamond bullet
BLT_SQUARE:               Square bullet
BLT_HOLLOW_SQUARE:       Hollow square bullet
BLT_4_DIAMONDS:          Four diamonds symbol
BLT_ARROW:                Arrow bullet
BLT_CHECK:                Check bullet
```

When the 'IsBullet' parameter is set to false, use one of the following constants for the 'type' parameter:

```
NBR_DEC:                  Decimal number
NBR_UPR_ALPHA             Uppercase Alphabetic
```

```

        number
        NBR_LWR_ALPHA Lowercase Alphabetic
        number
        NBR_UPR_ROMAN Uppercase Roman number
        NBR_LWR_ROMAN Lowercase Roman number

String TextBef;           // Text before the paragraph number (limited to 10 bytes).
                          // Set to null for default.

String TextAft;          // Text After the paragraph number (limited to one byte).
                          // Set to null for default.

int flags;                // Set this argument to BLTFLAG_HIDDEN to create a
                          // continuing list item. Set to 0 to create regular list item

boolean repaint;         // Repaint the screen after this operation

```

**Comment:** This method uses the older method of applying bullet and numbering. You can use the [TerSetListBullet](#) function to use the newer list mechanism to apply bullets and numbers. The new function has a better support for nested lists and multiple lists within a document.

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



## TerSetBulletId

**Assign a bullet id to a paragraph id.**

```
boolean TerSetBulletId( BulletId, Parald)
```

```
int BulletId;           // Bullet id to be assigned to the paragraph id
```

```
int Parald;             // Paragraph id
```

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

**See Also:**

[TerCreateBulletId](#)

[TerCreateParald](#)



## TerSetDefTabWidth



#### Set default tab width.

int TerSetDefTabWidth( NewWidth, repaint)

int NewWidth; // new tab width in twips

boolean repaint; // true to repaint the screen after this operation.

**Return Value:** This function returns the previous value of the tab width in twips.

#### See Also

[TerSetDefTabType](#)



### TerSetDefTabType

#### Set the default tab type.

boolean TerSetDefTabType(TabType)

int TabType; // Tab type: TAB\_LEFT, TAB\_RIGHT, TAB\_CENTER, TAB\_DECIMAL

**Description:** This function allows you to set the tab type for the left-mouse click on the ruler.

**Return Value:** This function returns True if successful

#### See Also

[TerSetDefTabWidth](#)



### TerSetParaAuxId

#### Set the Auxiliary id for the paragraph.

boolean TerSetParaAuxId( FirstLine, LastLine, AuxId)

int FirstLine; // The first line of the paragraph. Set this parameter to -1 to select the current paragraph or all paragraphs in the range of selected text (if any text selected)

int LastLine; // The last line for the paragraph. This argument is not used when 'FirstLine' is set to -1.

boolean AuxId; // The new auxiliary id for the paragraph.

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



## TerSetParaBkColor

**Set the background color for the paragraph.**

boolean TerSetParaBkColor( dialog, color, repaint)

boolean dialog; // true to show the dialog box for the user to select a background color. false to use the background color specified by the 'color' parameter.

Color color; // New background color for the paragraph.  
The new background color can also be specified as Html color string, such as "red", "blue", "#FF0000" .

boolean repaint; // true to repaint the screen after this operation.

**Return Value:** This function returns true when successful



## TerSetParald

**Set the paragraph id for the paragraph.**

boolean TerSetParald( FirstLine, LastLine, Parald)

int FirstLine; // The first line of the paragraph. Set this parameter to -1 to select the current paragraph or all paragraphs in the range of selected text (if any text selected).

int LastLine; // The last line for the paragraph. This argument is not used when 'FirstLine' is set to -1.

boolean Parald; // The new paragraph id for the paragraph.

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

**See Also;**  
[TerCreateParald](#)



## TerSetParaBorderColor

### Set the border color for the paragraph.

boolean TerSetParaBorderColor(color, repaint)

Color color; // New border color for the paragraph. This color is effective only if the paragraph borders are enabled. You can enable paragraph borders using the SetTerParaFmt function.

The new border color can also be specified as Html color string, such as "red", "blue", "#FF0000" .

boolean repaint; // True to repaint the screen after this operation.

**Return Value:** This function returns True when successful

### See Also

[SetTerParaFmt](#)



## TerSetParaList

### Set list numbering for the paragraph.

boolean TerSetParaList( dialog, Parald, ListOr, level, repaint)

boolean dialog; // Set to true to show a dialog box to the user to select list-override and level information.

boolean Parald; // The paragraph id to modify. Set to -1 to apply list numbering to the current paragraph, currently selected paragraph, or to the stylesheet item currently being edited.

int ListOr; // The list-override id to use for the paragraph.

int level; // The level number to use for the paragraph. A simple list allows only one list level (level 0). A nested list allows up to 9 levels (0 to 8).

boolean repaint; // Set to true to repaint the screen after this operation.

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

### See Also:

[TerCreateParald](#)  
[TerEditList](#)  
[TerEditListOr](#)  
[TerCreateListBullet](#)  
[TerSetListBullet](#)



## TerSetParaShading

**Set the shading value for the current paragraph.**

boolean TerSetParaShading( shading, refresh)

int shading; // shading amount (0 to 10000)

boolean refresh; // true to refresh the window after this operation.

**Description:** This function is used to specify the shading amount for the current paragraph or the range of selected paragraphs. The shading value of 10000 indicates the darkest shading, whereas the shading value 0 indicates no shading.

Return Value: This function returns true if successful.



## TerSetParaTextFlow

**Set the right-to-left/left-to-right text flow option for the paragraph.**

boolean TerSetParaTextFlow( dialog, TextFlow, refresh)

boolean dialog; // Set to true to show the user dialog.

int TextFlow; // The text flow constant can be one of the following:

FLOW\_LTR: Left-to-right text flow

FLOW\_RTL: Right-to-left text flow

FLOW\_DEF: Default text flow. The flow will be determined by the document, section or table level text flow specification.

boolean refresh; // true to refresh the window after this operation.

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

**See Also:**

[TerSetDocTextFlow](#)  
[TerSetSectTextFlow](#)  
[TerSetRowTextFlow](#)



## TerSetParaIndent

**Set the paragraph indentation.**

boolean TerSetParaIndent( left, right, first, repaint)

int left; // The left indentation in twips. Set this value to -1 to leave it unchanged

int right; // The right indentation in twips. Set this value to -1 to leave it unchanged.

int first; // The indentation adjustment (twips) to apply to the first line only. Set this value to -1 to leave it unchanged.

boolean repaint; // true to repaint the screen after this operation.

**Return Value:** The function returns true when successful.

**See Also:**

[ParaIndentTwips](#)  
[ParaLeftIndent](#)  
[ParaRightIndent](#)  
[ParaHangingIndent](#)



## TerSetParaSpacing

**Set the spacing parameters for the current paragraph.**

boolean TerSetParaSpacing( SpaceBefore, SpaceAfter, SpaceBetween, refresh)

boolean TerSetParaSpacing2( SpaceBefore, SpaceAfter, SpaceBetween, LineSpacing, refresh)

int SpaceBefore; // Space before the first line of the paragraph in twips  
Set to -1 to leave the previous value unchanged.

int SpaceAfter; // Space after the last line of the paragraph in twips.

Set to -1 to leave the previous value unchanged.

int SpaceBetween; // Minimum space between the lines of the paragraph. To set the exact spacing, specify a negative value. To set a minimum line spacing, specify a positive value.  
Set to -9999 to leave the previous value unchanged.

int LineSpacing: // Applicable to TerSetParaSpacing2 function only. You can use this argument to specify the extra line space in percentage of the current line height. For example, set to 50 to specify 1.5 line spacing, or 100 to specify double line spacing. Set to 0 for default.  
Set to -1 to leave the previous value unchanged.

boolean refresh; // true to refresh the window after this operation.

**Description:** This function is used to specify the paragraph spacing parameters. Use zero to specify the default value for any parameter.

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

#### See Also:

[ParaIndentTwips](#)  
[ParaLeftIndent](#)  
[ParaRightIndent](#)  
[ParaHangingIndent](#)



## TerSetPflags

### Set additional paragraph flags

boolean TerSetPflags( flags, OnOff, repaint)

int flags: Select paragraph flags:  
PFLAG\_NO\_WRAP: Disable word wrapping  
PFLAG\_WIDOW: Set Widow/Orphan control  
To specify more than one styles, use the 'logical OR' (|) operator.

boolean OnOff; // true to set the flags, false to reset the selected flags.

boolean repaint; // true to refresh the window after this operation

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

#### See Also:

[ParaNormal](#)

[SetTerParaFmt](#)



## TerSetTab

### Set a tab position:

```
boolean TerSetTab( TabType, TabPos, TabLeader, repaint)
```

```
int TabType;           // Tab type: TAB_LEFT, TAB_RIGHT, TAB_CENTER,
                       TAB_DECIMAL
```

```
int TabPos;           // Tab position (in twips unit) to create
```

```
BYTE TabLeader;      // Tab leader type:
                       TAB_NONE:   No tab leader
                       TAB_DOT:    Dotted line tab leader
                       TAB_HYPH:   Hyphen line tab leader
                       TAB_ULINE:  Underline tab leader
```

```
boolean repaint;     //Repaint the window after this
                       operation
```

**Description:** Use this function to create one tab position.

**Return Value:** This function returns true if successfulSee Also: [ClearTab](#), [ClearAllTabs](#)

### See Also:

[ClearTab](#)

[ClearAllTabs](#)



## Section Formatting

### In This Chapter

[TerColBreak](#)

[TerGetMarginEx](#)

[TerGetSectAlign](#)

[TerSetSectBorder](#)

[TerGetSectColWidth](#)

[TerGetSectBins](#)

[TerGetSectInfo](#)

[TerGetSectParam](#)

[TerGetPageOrient](#)

[TerGetSeqSect](#)

[TerSectBreak](#)

[TerSetMargin](#)

[TerSetPaper](#)

[TerSetSect](#)





```

int right;           // The variable to retrieve the right margin value in twip
                    // units. Example:
                    right=tej.TerGetOutInt("RightMargin");

int top;            // The variable to retrieve the top margin value in twip
                    // units.Example:
                    top=tej.TerGetOutInt("TopMargin");

int bottom;        // The variable to retrieve the bottom margin value in twip
                    // units. Example:
                    bottom=tej.TerGetOutInt("BotMargin");

int header;        // The variable to retrieve the distance of the header text
                    // from the top of the page. Example:
                    header=tej.TerGetOutInt("HeaderY");

int footer;        // The variable to retrieve the distance of the footer text
                    // from the bottom of the page. Example:
                    footer=tej.TerGetOutInt("FooterY");

```

**Return Value:** This function returns the total number of sections in the document if successful. Otherwise it returns 0.

#### See Also:

[TerSetMargin](#)



## TerGetSectAlign

**Retrieve the vertical alignment attribute for a section.**

```
int TerGetSectAlign(sect)
```

```

int sect;           // Section id to retrieve the page number format. You
                    // can also set this parameter to SECT_CUR to specify
                    // the current section.

```

**Return Value:** This function returns the alignment constant for the section. Please refer to the TerSetSectAlign function for the list of alignment constants.

#### See Also

[TerSetSectAlign](#)



## TerSetSectBorder

### Get the page border attribute for a section.

boolean TerGetSectBorder( sect, out type, out width, out space, out color)

int sect; // Section id to access. You can also set this parameter to SECT\_CUR to specify the current section.

**Upon a successful return from this method, the [TerGetOutInt](#) and related methods can be used to retrieve the temporarily saved values for these variables:**

int type; // Border type. It can be one of the following constants:

BRDRTYPE_SINGLE	Single line border
BRDRTYPE_DBL	Double line border
BRDRTYPE_TRIPLE	Triple line border
BRDRTYPE_SHADOW	Shadow border
BRDRTYPE_THICK_THIN	Thick-thin lines border
BRDRTYPE_THIN_THICK	Thin-thick lines border
BRDRTYPE_THICK_THIN_THICK	Thick-thin-thick border
BRDRTYPE_THIN_THICK_THIN	Thin-thick-thin border
BRDRTYPE_NONE	No Border

The following three parameters are not used when border type is BRDRTYPE\_NONE

To retrieve this value:

```
type=TerGetOutInt("BorderType");
```

int width; // The variable to retrieve the line thickness in twips units.

Example:

```
type=TerGetOutInt("BorderWidth");
```

int space; // The variable to retrieve the border distance from the edge of the page in twips units, Example:

```
type=TerGetOutInt("BorderSpace");
```

Color color; // The variable to retrieve the border color

**Return Value:** The function returns true when successful.

#### See Also

[TerSetSectBorder](#)



## TerGetSectColWidth

### Retrieve the column width or inter-column spacing for a variable width column section.

int TerGetSectColWidth(hWnd, sect, col, GetColWidth)

int sect; // Section id to access. You can also set this parameter to -1 to specify the current column.

int col; // The column number (0 to total columns -1) to retrieve the the width or column space values.

boolean GetColWidth; // Set to TRUE to return the column width for the specified column. Set to FALSE to return the space after the specified column.

**Return Value:** This function returns the column width or the space after the column for the specified column. The value is returned in twips unit. The function return -1 if an error occurs.



## TerGetSectBins

### Retrieve the paper bins used by a section.

boolean TerGetSectBins(sect)

int sect; // The section id to retrieve information. This parameter can assume a value between 0 and 'TotalSects-1'. It can also be set to SECT\_CUR to specify current section.

**Upon a successful return from this method, the TerGetOutMediaTray method can be used to retrieve temporarily saved values for these variables:**

MediaTray FirstPageBin; // The variable to retrieve the first page bin. To retrieve this value after call this method:

```
FirstPageBin=tej.TerGetOutMediaTray("FirstPageBin")
```

MediaTray NextPageBin; // The variable to retrieve the next page bin. To retrieve this value after call this method:

```
NextPageBin=tej.TerGetOutMediaTray("NextPageBin")
```

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

#### See Also

[TerGetSectInfo](#)

[TerSetSect](#)



## TerGetSectInfo

**Retrieve the current section parameters.**

```
boolean TerGetSectInfo()
```

**Upon a successful return from this method, the [TerGetOutInt](#) and [TerGetOutBool](#) methods can be used to retrieve the temporarily saved values for these variables:**

```
int NumCols;           // The variable to receive the number of columns for the section. To
                       // retrieve this value after calling this method:
                       NumCols=tej.TerGetOutInt("NumCols")

int ColSpace;          // The variable to receive the space between the columns in Twips.
                       // To retrieve this value after calling this method:
                       ColSpace=tej.TerGetOutInt("ColSpace")

boolean NewPage;       // This variable receives true if the section starts on a new page,
                       // otherwise it receives a false. To retrieve this value after calling this
                       // method:
                       NewPage=tej.TerGetOutBool("StartPage")

int FirstPageNo;      // This variable receives 0 if this section uses continuous page
                       // numbering, otherwise it receives the page number of the first page
                       // for this section. To retrieve this value after calling this method:
                       FirstPageNo=tej.TerGetOutInt("FirstPageNo")
```

**Return Value:** The function returns true when successful.

**See Also:**

[TerSetSect](#)



## TerGetSectParam

**Get the section parameters.**

```
int TerGetSectParam(hWnd, id, type)
```

```
int id;                // Section id (0 to Total Sections-1) to retrieve parameters.
```

```
int type;              // The parameter to retrieve:
```

**SP\_FLAGS:** Returns the flags bits associated with the section id. Please use the AND parameter with the return value to check if one of the following flags is applicable:

**SECT\_NEW\_PAGE:** Start the section on a new page.

**SECT\_RESTART\_PAGE\_NO** Restart page numbers.

**SECT\_VALIGN\_CTR** Vertically center align the page text.

**SECT\_VALIGN\_BOT** Vertically bottom align the page text.

**SECT\_LINE** Section uses line numbering.

**SECT\_SNAP\_LINE\_GRID** Text within the section is aligned to a grid.

**SP\_GUTTER\_MARG** Gutter margin in twips.

**SP\_LINE\_STEP:** Returns the steps in which the line numbering is applied when the SECT\_LINE flag (see SP\_FLAGS) is applied. A value of 0 or 1 indicates continuous line numbering.

**Return Value:** The function returns the value for the requested parameter. It returns FP\_ERROR to indicate an error condition.



## TerGetPageOrient

**Get the page orientation and dimensions.**

boolean TerGetPageOrient( PageNum)

boolean TerGetPageOrientEx( PageNum)

boolean TerGetPageOrient2( PageNum)

```
int PageNum; // Page number between 0 and TotalPages-1
```

**Upon a successful return from this method, the [TerGetOutInt](#) method can be used to retrieve the temporarily saved values for these variables:**

```
int pWidth; // Variable to receive the page width (in twips) after  
            // considering the orientation. To retrieve this value after  
            // calling this method:
```

```
width=tej.TerGetOutInt("Width");
```

```
int pHeight; // Variable to receive the page height (in twips) after  
            // considering the orientation. To retrieve this value after  
            // calling this method:
```

```
height=tej.TerGetOutInt("Height");
```

```
int pHiddenX; // Variable to receive the printer hidden area (in twips) in  
            // the x direction. To retrieve this value after calling this  
            // method:
```

```
HiddenX=tej.TerGetOutInt("HiddenX");
```

```
int pHiddenY; // Variable to receive the printer hidden area (in twips) in  
            // the y direction. To retrieve this value after calling this  
            // method:
```

```
HiddenY=tej.TerGetOutInt("HiddenY");
```

**Description:** This function is available in the PageMode or PrintView mode only.

**Return Value:** This function returns true if the page uses the portrait orientation. Otherwise, it returns a false value.



## TerGetSeqSect

**Translate a section id into the sequential section number.**

```
int TerGetSeqSect( SectId)
```

```
int SectId; // Section id for the text.
```

Description: This function translates the section id into the sequential section numbers. Please note that the section id assigned to the text are not sequential. For example, in a document containing 3 sections, it is not correct to assume id 0 for the first section, or id 1 for the subsequent section of the document. Most APIs involving section need you to specify the section id. A section id can be retrieved using the GetTerFields or TerGetPageSect functions. However, certain functions such as TerPosBodyText and TerPosHdrFtr need the sequential section number for the section argument. For the purpose of differentiation, this manual uses the term 'section id' or 'sequential section

number' as appropriate.

**Return Value:** The function returns the sequential section number when successful. Otherwise it returns -1.

**See Also:**

[TerPosBodyText](#)

[TerPosHdrFtr](#)

[TerGetPageSect](#)



## TerSectBreak

**Create a new section.**

boolean TerSectBreak(repaint)

boolean repaint;                      //Repaint the window after this operation

**Description:** This function is used to place the following text on the new section. The section break is created before the current line. If you have enabled the editing of header/footer text, please turn it off before calling this function. Please note that a section break can not be created inside an object such as table, frame, text box, etc.

A section break is indicated by a double solid line.

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

**See Also:**

[TerPageBreak](#)

[TerColBreak](#)



## TerSetMargin

**Set the margin values for the sections in the document.**

boolean TerSetMargin(left,right,top,bottom,repaint)

boolean TerSetMarginEx(sect,left,right,top,bottom,header,footer,repaint)

int sect;                                      // The section id to modify. This parameter can assume a value between 0 and 'TotalSects-1'. It can also be set to SECT\_CUR to modify the current section only, or it can be set to SECT\_ALL to modify all sections in the current document. The TerSetMargin function implicitly uses a value of SECT\_CUR for this parameter

int left;                                      // The left margin value in twip units. Set to -1 to leave this parameter unchanged.

int right;	// The right margin value in twip units. Set to -1 to leave this parameter unchanged.
int top;	// The top margin value in twip units. Set to -1 to leave this parameter unchanged.
int bottom;	// The bottom margin value in twip units. Set to -1 to leave this parameter unchanged.
int header;	// The distance of the header text from the top of the page. Set to -1 to leave this parameter unchanged.
int footer;	// The distance of the footer text from the bottom of the page. Set to -1 to leave this parameter unchanged.
boolean repaint;	// set to true to repaint the screen after this operation

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

**See Also:**

[TerGetMarginEx](#)



## TerSetPaper

**Set custom paper size and orientation.**

boolean TerSetPaper( width, height, IsPortrait, refresh) // html/javascript version

boolean TerSetPaper( size, IsPortrait, refresh) // java version

int width;	// new paper width in twips (1440 twips = 1 inch)
------------	---

int height;	// new paper height in twips.
-------------	-------------------------------

PaperSize size;	// Paper size
-----------------	---------------

boolean IsPortrait	// Set to true to specify the portrait orientation.
--------------------	---

boolean refresh;	// true to refresh the window after this operation.
------------------	---

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



## TerSetSect



### **Set section parameters.**

boolean TerSetSect( NumCols, ColSpace, NewPage)

boolean TerSetSectEx( NumCols, ColSpace, NewPage, FirstPageNo)

boolean TerSetSect2( NumCols, ColSpace, NewPage, FirstPageNo, FirstPageBin, NextPageBin)

boolean TerSetSect3( NumCols, ColSpace, NewPage, SetBins, FirstPageNo, FirstPageBin, NextPageBin, SectId, size)

boolean TerSetSect3( NumCols, ColSpace, NewPage, SetBins, FirstPageNo, FirstPageBin, NextPageBin, SectId, size, IsPortrait)

int NumCols; // number of columns for the section. Set to 0 to show a user dialog. Set to -1 to leave this value unchanged.

int ColSpace; // Space between the columns in Twips.

boolean NewPage; // true to begin the section on a new page.

int FirstPageNo; // Page number for the first page of the current section. Set to 0 to use default page numbering.

boolean SetBins; // Set to true to apply the FirstPageBin and NextPageBin parameters.

int FirstPageBin; / Bin selection to print the first page of this section. Set to MediaTray enumeration.

int NextPageBin; // Bin selection to print the subsequent pages of this section. Set to MediaTray enumeration.

int SectId; // Section id (0 to TotalSect-1). Set to -1 to use the current section.

int size; // Not used. Set to 0.

boolean IsPortrait; // Set to true to use the portrait orientation. set to false to apply landscape orientation.

### **Example:**

```
tej.TerSetSect(2, 720, true);
```

The above statement sets the current section to use 2 column layout.

```
tej.TerSetSect3(1, 0, true, 1, false,
                MediaTray.MAIN,
                MediaTray.MAIN,
                0, 0,
                false);
```

The above statement modifies section id 0. It applies the landscape orientation. The paper bin application is disabled by setting the SetBins to false.

**Return Value:** The function returns true when successful.

**See Also:**

[TerSetSectOrient](#)

[TerGetSectInfo](#)



## TerSetSectAlign

**Set the vertical alignment for the section.**

boolean TerSetSectAlign(sect, align, refresh)

```
int sect; // Section id to apply changes. You can also set this
          // parameter to SECT_CUR to edit the current section, or
          // set it to SECT_ALL to apply changes to all sections in
          // the document.

int align; // The 'align' parameter can be set to one of the
           // following constants:

           SECT_VALIGN_CTR  Center vertically

           SECT_VALIGN_BOT  Bottom aligned page

           0                 Top aligned page (default)

boolean refresh; // TRUE to refresh the window after this operation
```

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



## TerSetSectBorder

**Set the page border for a section.**

boolean TerSetSectBorder( sect, type, width, space, color, repaint)

```
int sect; // Section id to apply changes. You can also set this
          // parameter to SECT_CUR to edit the current section, or
```

set it to SECT\_ALL to apply changes to all sections in the document.

```
int type; // Border type. It can be one of the following constants:

BRDRTYPE_SINGLE      Single line border
BRDRTYPE_DBL         Double line border
BRDRTYPE_TRIPLE      Triple line border
BRDRTYPE_SHADOW      Shadow border
BRDRTYPE_THICK_THIN  Thick-thin lines border
BRDRTYPE_THIN_THICK  Thin-thick lines border
BRDRTYPE_THICK_THIN_THICK  Thick-thin-thick border
BRDRTYPE_THIN_THICK_THIN  Thin-thick-thin border
BRDRTYPE_NONE        No Border

int width; // Line thickness in twips units

int space; // Border distance from the edge of the page in twips units

Color color; // Border color

// The new border color can also be specified as Html color
// string, such as "red","blue", "#FF0000"

boolean repaint; // true to refresh screen after this operation.
```

**Return Value:** The function returns true when successful.

#### See Also:

[TerSetSect](#)

[TerSetSectBorder](#)



## TerSetSectColWidth

**Set the column width and inter-column spacing for a variable width column section.**

```
boolean TerSetSectColWidth(sect, col, width, ColSpace, repaint)
```

```
int sect; // Section id to apply changes. You can also set this
// parameter to -1 to specify the current column.
```

```
int col; // The column number (0 to total columns -1) to apply
// the width and column space parameters. Set this
// parameter to -1 to restore the section to use uniform
// width for the columns.
```

```
int width; // The new width (int twips) for the specified column.

int ColSpace; // The new inter-column space (in twips) after the
              // specified column.

boolean repaint; // TRUE to refresh the window after this operation
```

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



## TerSetSectLineNbr

**Set line numbering for the section.**

```
boolean TerSetSectLineNbr(sect, set, repaint)
```

```
boolean TerSetSectLineNbr2(sect, set, step, repaint)
```

```
int sect; // Section id to apply changes. You can also set this
          // parameter to SECT_CUR to edit the current section, or
          // set it to SECT_ALL to apply changes to all sections in
          // the document.
```

```
boolean set; // Set to TRUE to enable line numbering. Set to false to
            // disable line numbering.
```

```
int step; // Steps in which to draw the numbers. A value of 0 or 1
          // produces continuous line numbering.
```

```
boolean repaint; // TRUE to refresh the window after this operation
```

**Comment:** The line numbers are displayed on the left side of the page. The page-layout (ID\_SHOW\_PAGE\_LAYOUT) display must be turned on to see line numbering.

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



## TerSetSectOrient

**Set the orientation for a section.**

```
boolean TerSetSectOrient( orient, repaint)
```

```
boolean IsPortrait; // Set to true to set to portrait orientation.
```

Otherwise set to false.

```
boolean repaint;           // true to refresh screen after this operation.
```

**Description:** This function is used to set the page orientation for the section..

**Return Value:** The function returns true when successful.

**See Also:**

[TerSetSect](#)



## TerSetSectPageSize

**Set the page size for a section.**

```
boolean TerSetSectPageSize( sect, kind, PageWidth, PageHeight, repaint)
```

```
int sect;                  // Section id to apply changes. You can also set this
                           // parameter to SECT_CUR to edit the current section, or
                           // set it to SECT_ALL to apply changes to all sections in
                           // the document.
```

```
int kind;                  // Not used currently. Must be set to 0.
```

```
int PageWidth;            // The page width in twips units.
```

```
int PageHeight;          // The page height in twips units.
```

```
boolean repaint;         // true to refresh screen after this operation.
```

**Return Value:** The function returns true when successful.

**See Also:**

[TerSetSect](#)



## TerSetSectParam

**Set section parameters.**

```
boolean TerSetSectParam(hWnd, select, type, val, repaint)
```

```
int select;               // Section id to apply changes. You can also set this
                           // parameter to SECT_CUR to edit the current section, or
                           // set it to SECT_ALL to apply changes to all sections in the
                           // document.
```

int type;	Parameter type to set. Select one of the constants:
SP_LINE_BET_COL	Set to 1 to draw a line between columns. Applicable to a multi-column section only.
SP_LEFT_MARG	Get the section left margin value in twips.
SP_RIGHT_MARG	Get the section right margin value in twips.
SP_TOP_MARG	Get the section top margin value in twips.
SP_BOT_MARG	Get the section bottom margin value in twips.
SP_HDR_MARG	Get the distance of the header text from the top of the page in twips.
SP_FTR_MARG	Get the distance of the footer text from the bottom of the page in twips.

int val;                      New value for the parameter specified by the 'type' parameter.

boolean repaint;              TRUE to repaint the screen after this operation

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



## TerSetSectTextFlow

**Set the right-to-left/left-to-right text flow option for the section.**

boolean TerSetSectTextFlow( sect, TextFlow, refresh)

int sect;                      // Section id to apply changes. You can also set this parameter to SECT\_CUR to edit the current section, or set it to SECT\_ALL to apply changes to all sections in the document.

int TextFlow;                 // The text flow constant can be one of the following

FLOW\_LTR                      Left-to-right text flow

FLOW\_RTL                      Right-to-left text flow

FLOW\_DEF

Default text flow. The flow will be determined by the document, table, or paragraph level text flow specification.

boolean refresh;

// true to refresh the window after this operation

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

#### See Also:

[TerSetDocTextFlow](#)

[TerSetParaTextFlow](#)

[TerSetRowTextFlow](#)



## Document

### In This Chapter

[GetTerFields](#)

[SetTerFields](#)

[TerGetParam](#)

[TerGetRtfDocInfo](#)

[TerGetWordCount](#)

[TerInsertDateTime](#)

[TerInsertToc](#)

[TerInsertToc2](#)

[TerLoadExtFont](#)

[TerSetDefDir](#)

[TerSetDocTextFlow](#)

[TerSetRtfDocInfo](#)



## GetTerFields

### Retrieve Window Variables:

TejTerField GetTerFields()

Description: This function returns various operational parameters for the current TEJ window.

**This function should be used sparingly since it is resource intensive. In most instances you can retrieve the same information using other methods.**

```
class TejTerField {
```

The following fields are read/write fields. To update a field you must retrieve the current values by calling the *GetTerFields* function. Modify the fields that you wish to, and then call the *SetTerFields* function to make the new value effective.

int size;	The size of this structure in number of bytes.
int CurCol	Current window column position (0 to one less than the length of the line).
int PaintEnabled	A false value disables the screen painting and word wrapping until it is re-enabled using another call to the <i>SetTerFields</i> function
int WrapFlag	Wrap control. This function can be used to temporarily suspend word wrapping. The word wrapping is automatically enabled when the user hits a keystroke or makes a selection from the menu.
int CurRow	Current window row position ( 0 to the height of the window). This field is not meaningful in Page mode or Fitted View modes.
int BeginLine	First line number in the window. The editor ensures that CurRow is always equal to CurLine minus BeginLine (this field is not meaningful in Page mode or Fittend View modes).
int CurLine	Current line number in the file (0 to one less than the total number of lines in the file).
Color TextBkColor	Background color for the window.
Color StatusBkColor	Background color of the status line
Color StatusColor	Foreground color of the status line
int HilightType	Line or character highlighting flag (see HILIGHT_ constants in the TEJ.H file). Use this flag and the following variables to set or reset text selection.
int HilightBegCol	Beginning column number of the highlighted block
int HilightEndCol	Ending column number of the highlighted block.
int HilightBegRow	Beginning line number of the highlighted block
int HilightEndRow	Ending line number of the highlighted block
boolean StretchHilight	A true value allows the user to stretch the current highlighted block by using the mouse or arrow keys.

Please do not use the following two fields as they are being phased out. Use the



TerGetLine function to retrieve the text and font ids for a line number.

char[] text	Text data for the current line.
ushort[] font	Font id for every character in the 'text' array. Use the 'GetFontInfo' function to get further information about an editor font id.
int pfmt	Paragraph id of the current line
int LineLen	Length of the current line
int TextApply	Use this variable to specify how the 'text' and 'font' data should be applied to the current TEJ window, see APPLY_ constants in the TEJ.H file. Using this flag you can modify the current line, or insert a new line after or before the current line.
boolean Reclamesources	true to reuse unused font and paragraph ids.
boolean ModifyProtectColor	true to show the protected text in a lighter shade
boolean LinkDbClick	true to fire hyperlink on mouse double click. Otherwise single click is used to fire the hyperlink event.
boolean ShowProtectCaret	true to display caret even when positioned on protected text.
int LinkStyle	The character style of the hyperlink phrase. When this style is set to HLINK, then the following LinkColorW variable is not used for detecting a link.
Color LinkColor	The color of the hyperlink phrase.
boolean SnapToGrid	true to snap tabs and margin on the ruler to an invisible grid
boolean HtmlMode	true to enable html mode adjustments
boolean ShowTableGridLines	true to show table grid lines
The following are the read only fields. TEJ will ignore any modification to these fields.	
IntPtr hTerWnd	Handle to the editor window
Graphics TerGr	Handle to TEJ class DC. Call the TerGetBufferDC function if you wish to retrieve the handle of the associated buffer device context.

Rectangle TerRect	Entire client window rectangle
Rectangle TerWinRect	Text window rectangle
int TotalLines	Total lines in the file
int MouseLine	Current text line position of the mouse pointer. Current row position is given by MouseLine minus BeginLine.
int MaxColBlock	Biggest column block allowed
int TotalPfmts	Total paragraph ids in use by the current window.
int TotalFonts	Total font objects in use by the current window
int TotalStyles	Total number of stylesheet items
int WinWidth	Current window width in character columns
int WinHeight	Number of lines displayed in the window. This field is not meaningful in Page mode or Fitted View modes.
int TerWinOrgX	Window origin x co-ordinates used to set the view port
int MouseCol	Current text column position of the mouse pointer.
boolean modified	Data modified, user needs to select the 'save' option to save data
boolean WordWrap	True when the word wrap is turned on
int ParaLeftIndent	Paragraph left indent in twips
int ParaRightIndent	Paragraph right indent in twips
int ParaFirstIndent	Paragraph first line indent in twips
int ParaFlags	Paragraph flags. Refer to the SetTerParaFmt function for a list of paragraph flags.
int ParaTabId	Paragraph tab id (index into the tab table)
int ParaCellId	Paragraph cell id (index into the cell table)
int ParaShading	Paragraph shading (0 to 10000)
int ParaSpaceBefore	Space before the paragraph in twips

int ParaSpaceAfter	Space after the paragraph in twips
int ParaSpaceBetween	Minimum space between
int ParaStyleId	Paragraph style Id
int ParaAuxId	Paragraph aux id
int pflags	Paragraph PFLAG_ flag constants
int CurSect	The section id of the current line. You can use the TerGetSeqSect function to translate the section id into the sequential section number.
int LeftMargin	Section left margin in twips
int RightMargin	Section right margin in twips
int TopMargin	Section top margin in twips
int BotMargin	Section bottom margin in twips
int columns	Number of columns in the current section
int CurPage	Current page number
int TotalPages	Number of pages in the document
int MouseX	Recent mouse click x position
int MouseY	Recent mouse click y position
boolean PrintView	true when the print view mode is turned on
boolean PageMode	true when the page mode is turned on
boolean FittedView	true when the fitted view mode is turned on
boolean ShowParaMark	true when showing paragraph markers
boolean ShowHiddenText	true when showing the hidden text
int CurCtlId	Currently selected control id
int ParaFrameFlags	Current paragraph frame flags (PARA_FRAME_? constants defined in the tej.h file)

**Return Value:** A true value indicates success.

### See Also

[SetTerFields](#)

[GetFontInfo](#)



## SetTerFields

### Set Window Variables:

```
boolean SetTerFields(field)
```

```
class TejTerField *field;          // information buffer (see the GetTerFields function)
```

**Description:** This function sets various operational parameters for the current TEJ window. You must first call the GetTerFields function to retrieve the current values of the parameters. You can then change the variables that you need to change. The TEJ editor validates the information before applying them to the current window.

**Return Value:** A true value indicates success.

### See Also:

[TerGetField](#)

[TerSetCtlColor](#)



## TerGetParam

### Retrieve miscellaneous operating variables.

```
boolean TerGetParam( type)
```

```
int type;          // The parameter type to retrieve:
```

```
TP_CUR_LINE          Current line number
```

```
TP_CUR_COL          Current column number
```

```
TP_CUR_SECT          Current section id
```

```
TP_MOUSE_X          Return the pixel x position where  
                    the mouse is positioned.
```

This value is relevant only when retrieved from the PreProcess or Action events for a mouse message.

TP_MOUSE_Y	<p>Return the pixel y position where the mouse is positioned.</p> <p>This value is relevant only when retrieved from the PreProcess or Action events for a mouse message.</p>
TP_MOUSE_LINE	<p>Return the text line number where the mouse is positioned.</p> <p>This value is relevant only when retrieved from the PreProcess or Action events for a mouse message.</p>
TP_MOUSE_COL	<p>Return the text column number where the mouse is positioned.</p> <p>This value is relevant only when retrieved from the PreProcess or Action events for a mouse message.</p>
TP_MOUSE_FONT_ID	<p>Return the font-id for the character where the mouse is positioned.</p> <p>This value is relevant only when retrieved from the PreProcess or Action events for a mouse message.</p>
TP_MOUSE_PICT_ID	<p>Return the picture-id for the picture where the mouse is located. This value would be 0 if the mouse is not positioned over a picture object.</p> <p>This value is relevant only when retrieved from the PreProcess or Action events for a mouse message.</p>
TP_MOUSE_FIELD_ID	<p>Return the field-id for the field where the mouse is positioned. This value would be 0 if the mouse is not positioned over a picture.</p> <p>This value is relevant only when retrieved from the PreProcess or Action events for a mouse message.</p>
TP_MOUSE_ON_TEXT_LINE	<p>Returns TRUE if the mouse is</p>

located on a text line. This value is meaningful only when retrieved after calling the TerPixToTextPos method.

TP_PAGE_BK_COLOR	Page background color
TP_SELECTION_TYPE	Selection type: Set to HIGHLIGHT_OFF if no text is selected. Any other value indicates that a text block is selected.  <i>The following four TP_SELECTION_ constants are not valid if the selection type is HIGHLIGHT_OFF.</i>
TP_SELECTION_START_LINE	Selection start line.
TP_SELECTION_START_COL	Selection start column.
TP_SELECTION_END_LINE	Selection end line.
TP_SELECTION_END_COL	Selection end column.
TP_TOTAL_BLTS	Total number of bullet ids in the document.
TP_TOTAL_CELLS	Total number of cell ids in the document.
TP_TOTAL_CHAR_TAGS	Total number of character tags in the document
TP_TOTAL_FONTS	Total number of font ids in the document.
TP_TOTAL_IMAGE_MAPS	Total number of image maps in the document.
TP_TOTAL_LINES	Total number of lines in the document.
TP_TOTAL_LISTS	Total number of list ids in the document.
TP_TOTAL_LIST_OR	Total number of list-override ids in the document.
TP_TOTAL_PAGES	Total number pages in the

	document.
TP_TOTAL_PARA_FRAMES	Total number of paragraph frames in the document
TP_TOTAL_PFMTS	Total number of paragraph ids in the document.
TP_TOTAL_SECTS	Total number of section ids in the document.
TP_TOTAL_STYLES	Total number of style ids in the document.
TP_TOTAL_TABS	Total number of tab ids in the document.
TP_TOTAL_TABLE_ROWS	Total number of table row ids in the document.
TP_WATERMARK_WASH	The return value is 1 if watermark picture is washed, otherwise the return value is 0.
TP_WATERMARK_PICT	Returns the picture id for the current watermark picture. Returns 0 if watermark is not set for the document
TP_TOTAL_REVIEWERS	Total number of reviewer ids. Id# 0 is not used.

**Return Value:** This function returns the value of the requested parameter. It returns -1 to indicate an error condition.

**See Also:**

[TerSetPictInfo](#)  
[TerPastePicture](#)  
[TerInsertPictureFile](#)  
[TerPictureFromFile](#)  
[TerGetPictOffset](#)



**TerGetRtfDocInfo**

Retrieve the header information about the document.

```
int TerGetRtfDocInfo( type, text)
```

```
int type; // Information type. Use one of the following constants:
```

```
INFO_TITLE Document title
```

```
INFO_SUBJECT Subject
```

```
INFO_AUTHOR Author
```

```
INFO_MANAGER Manager
```

```
INFO_COMPANY Company
```

```
INFO_OPERATOR Operator
```

```
INFO_CATEGORY Category
```

```
INFO_KEYWORDS Keywords
```

```
INFO_COMMENT Comment
```

```
INFO_DOCCOMM Additional comment
```

```
INFO_HLINKBASE Hyperlink base path
```

```
String text; // (output) Information text. Set to null to simply retrieve the length of the information text.
```

**Return Value:** When successful, this function returns the length of the information text, otherwise it returns 0. It also returns the information text using the 'text' argument.

#### See Also:

[TerSetRtfDocInfo](#)



## TerGetWordCount

**Count number of words in the document.**

```
int TerGetWordCount( flags)
```

```
int flags; // The flag can be one or more of the following bits:
```

```
WC_SELECTION: Scan only the selected text.
```



If a text block is not highlighted or if the WC\_SELECTION bit is not specified, then the entire document will be scanned.

WC\_INCLUDE\_HIDDEN: Count the hidden words also.

WC\_INCLUDE\_HDR\_FTR: Count the words in the header/footer area also.

**Return Value:** This function returns the number of words counted. It returns -1 to indicate an error condition.



## TerInsertDateTime

### Insert a date/time field.

boolean TerInsertDateTime( format, repaint)

String format; // Date time format (see description). Set this parameter to null to display a date format selection dialog box.

boolean repaint; //Repaint the window after this operation

**Description:** The format argument accepts a format string. A format String consists of day, date, month, year, second, hour and delimiter components. Example:

```
TerInsertDateTime( "M/d/yy" ,true);
```

```
TerInsertDateTime( "M-d-yy " ,true);
```

```
TerInsertDateTime( "d/M/yy" ,true);
```

Following is a list of various format components using an example date: June 8, 1999, 2:30:01 PM

Format

String Component Example

-----

D day 8

M month 6

dd day, 0 padded 08

MM month, 0 padded 06

yy year, 2 digits 99

yyyy year, 4 digits 1999  
ddd day (abbr) Tue  
MMM month (abbr) Jun  
dddd day Tuesday  
MMMM month June  
HH hour, 24 hour format 14  
mm minutes 30  
ss seconds 01  
h 12 hour format 2  
hh 12 hour format, 0 padded 02  
am/pm AM or PM am/pm  
Example date format strings:  
"d MMM yy" 8 June 99  
"dd/MM/yyyy " 08/06/1999  
"dddd, d MMMM yyyy" Tuesday, 8 June 1999  
" h:mm" 2:30  
"HH:mm" 14:30  
"hh:mm:ss am/pm" 02:30:01 PM  
"dddd, d MMMM yyyy hh:mm:ss am/pm" Tuesday, 8 June 1999 02:30:01 PM

Note: Date format is case-sensitive.

**Delimiter:** A delimiter may be used to separate the date components. The delimiter could be '/', '-', comma, spaces or any character not used by the date components.

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



## TerInsertToc

**Insert table of contents.**

boolean TerInsertToc( repaint)

boolean repaint; //Repaint the window after this operation

**Description:** This function scans the document to build a table of contents at the current cursor location. It includes a text line in the table of contents if it uses a paragraph style and the name of the paragraph style is in the form of 'heading n', when 'n' is a number from 1 to 9. The heading number is used to specify the indentation level. This function uses paragraph styles 'toc n' for the assembled heading lines. The top level heading (heading 1) is assigned the style 'toc 1', and so on. The editor would automatically create

any missing 'toc' style.

To insert a table of contents, first create the heading styles using the `TerEditStyle` function. For example, if you wish to insert a three level deep table of contents, create heading styles 'heading 1', 'heading 2', and 'heading 3'. Then place the cursor at the heading lines and apply a suitable heading style using the `TerSelectParaStyle` function. The last step would be to position the cursor where you wish to insert the table of contents and call the `TerInsertToc` function.

The table of contents are automatically updated whenever repagination occurs.

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

#### See Also:

[TerEditStyle](#)

[TerSelectParaStyle](#)

[TerInsertPageRef](#)



## TerInsertToc2

### Insert table of contents.

boolean `TerInsertToc2(TocType, styles, MinLevel, MaxLevel, repaint)`

int `TocType`; // The table-of-contents construction method. Use one of the following types:

`TOC_HEADINGS`: Construct the table-of-contents with the text using the heading styles named following the format 'heading n', where 'n' is a number from 1 to 9.

`TOC_OUTLINES`: Construct the table-of-contents with the text using the style with a outline level 0 to 8.

`TOC_CUSTOM`: Construct the table-of-contents with the text using the custom styles specified in the 'styles' parameter.

`TOC_FIELD`: Construct the table-of-contents with the text using the 'tc' field. The 'tc' field can be applied to the text using the [TerSetTcField](#) function.

String `styles`; // The list of custom styles to use to construct the table of contents. Each style name in this list must be delimited using the comma delimiter.

This parameter is used only when the `TocType` parameter is set to `TOC_CUSTOM`.

int `MinLevel`; // The minimum heading level to use.

int `MaxLevel`; // The maximum heading level to use.

boolean repaint; // Repaint the window after this operation

**Description:** This function provides additional flexibility for creating the table-of-contents than the simpler method called [TerCreateTej](#).

The table of contents are automatically updated whenever repagination occurs.

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



## TerLoadExtFont

**Load an external font file.**

boolean TerLoadExtFont(typeface, FontFile, type)

String typeface; // Actual typeface of the font as specified in the font-file.

String FontFile // Path or name of the FontFile containing the font data.

boolean UpdateToolbar // Set to true to update the toolbar immediately.

**Comment:** You can call this method more than once to load multiple fonts. The fonts loaded by this method are effective only during the session.

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

**Example:**

```
TerLoadExtFont( "Pirulen", "pirulen.ttf", true );
```



## TerSetDefDir

**Set the default directory and file type.**

boolean TerSetDefDir( dir, type)

String dir; // The default directory. Set to "" to use the program directory.

int type; // Input file type:

SAVE_RTF:	RTF file
SAVE_TEXT:	Text file
SAVE_UTEXT:	Unicode Text Format (not available in the 16 bit product)

**Description:** This function is used to set the initial directory and the file type display by the File Open dialog.

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

#### See Also

[TerSetLinkPictDir](#)



## TerSetDocTextFlow

**Set the right-to-left/left-to-right text flow option for the document.**

boolean TerSetDocTextFlow( dialog, TextFlow, refresh)

boolean dialog; // Set to true to show the user dialog.

int TextFlow; // The text flow constant can be one of the following:

FLOW\_LTR: Left-to-right text flow

FLOW\_RTL Right-to-left text flow

FLOW\_DEF Default text flow. The flow will be determined by the , section, table, or paragraph level text flow specification.

boolean refresh; // true to refresh the window after this operation.

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

#### See Also:

[TerSetParaTextFlow](#)

[TerSetSectTextFlow](#)

[TerSetRowTextFlow](#)



## TerSetRtfDocInfo

### Set the header information about the document.

boolean TerSetRtfDocInfo( type, text)

int type; // Information type. Please refer to the TerGetRtfDocInfo function for the constant values for this argument.

String text; // Information text to set.

**Comment:** Information text specified by this function is saved with the document, only if the document is saved in the RTF format. This information is not saved in the native format document.

**Return Value:** This function returns true when successful, otherwise it returns false.

#### See Also:

[TerGetRtfDocInfo](#)



## Text Selection

### In This Chapter

[DeselectTerText](#)

[SelectTerText](#)

[TerGetSelection](#)

[TerGetTextSel](#)

[TerLineSelected](#)

[TerNormalizeBlock](#)



## DeselectTerText

### Deselect previous selected text

boolean DeselectTerText( repaint)

boolean repaint; // true to refresh the window after this operation.

**Description:** Use this function to deselect previously selected text.

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

#### See Also

[SelectTerText](#)



## SelectTerText



```
int FirstCol;           // Beginning column number of the block. To retrieve this
                        // value after calling this method:
                        FirstCol=tej.TerGetOutInt("BegCol")

int EndLine;           // Last line number of the block. To retrieve this value
                        // after calling this method:
                        EndLine=tej.TerGetOutInt("EndLine")

int EndCol;           // The first non-selected column position. The ending
                        // column position is not included in the selected block.
                        // To retrieve this value after calling this method:
                        EndCol=tej.TerGetOutInt("EndCol")
```

**Return Value:** This function returns true if a text block is selected, otherwise it returns false.

#### See Also:

[DeselectTerText](#)  
[SelectTerText](#)  
[TerIsTableSelected](#)



## TerGetTextSel

**Retrieve the selected text.**

```
String TerGetTextSel()
```

**Return Value:** This function returns the String containing the selected data as plain text (excluding any hidden text).

A null value of the handle indicates an error.

#### See Also:

[TerGetRtfSel](#)  
[GetTerBuffer](#)  
[SetTerBuffer](#)  
[ReadTerFile](#)  
[SaveTerFile](#)  
[TerSearchReplace](#)



## TerLineSelected

**Check if a text line is selected in the highlighted block of text.**

```
boolean TerLineSelected( LineNo)
```



```
int LineNo;                // text line number.
```

**Return Value:** This function returns true if a text block is highlighted in the control and the given line is also highlighted. It **also** returns true if no text block is highlighted in the editor. It returns false if a text block is highlighted but the given line is not.



## TerNormalizeBlock

**Normalize selected text block.**

```
boolean TerNormalizeBlock()
```

**Description:** This function is used to adjust the beginning and ending of a selected text block, such that the beginning position is smaller than the ending position.



## Cursor and Text Position

### In This Chapter

[GetTerCursorPos](#)

[SetTerCursorPos](#)

[TerAbsToRowCol](#)

[TerEngageCaret](#)

[TerGetCaretPos](#)

[TerGetVisibleCol](#)

[TerPixToTextPos](#)

[TerPosLineAtTop](#)

[TerPosBodyText](#)

[TerRowColToAbs](#)

[TerScrToTwipsX](#)

[TerScrToTwipsY](#)

[TerSetCaretPos](#)

[TerTextPosToPix](#)



## GetTerCursorPos

**Retrieve the current cursor position.**

```
boolean GetTerCursorPos(CurCol)
```

```
int CurCol;                // The integer variable where the current column number  
                           // is returned.
```

Input: Set 0 to return the line/col pair of values. Set to -1 to return the current character position using the CurLine value.

Output: When the CurCol input value is set to 0, after a successful return from this method, you can retrieve the the current-column value using:

```
if (tej.GetTerCursorPos(CurCol) {  
    CurCol=tej.TerGetOutInt("Col");  
}
```

```
int CurLine;
```

// After a successful return from this method, you can retrieve the the current-line value using:

```
if (tej.GetTerCursorPos(CurCol) {  
    CurLine=TerGetOutInt("Line");  
}
```

Description: This function returns the current cursor position. The cursor position can be retrieved as the absolute position or in terms of the line number and column number. To get the absolute cursor position, set the CurCol variable to -1 before calling this function. The absolute position (base 0) is returned in the CurLine variable.

Example: int CurCol=-1; int CurLine;

**GetTerCursorPos(CurCol);** // returns absolute position in the CurLine variable.

```
if (tej.GetTerCursorPos(CurCol) {  
    CurLine=TerGetOutInt("Line");  
}
```

To get the line (base 0) and column (base 0) position of the cursor, set the CurCol variable to a value other than -1 before calling this function.

```
int CurCol=0;
```

```
int CurLine;
```

```
GetTerCursorPos(CurCol);
```

// The current line number is returned in the CurLine variable, the current column is returned in the CurCol variable.

```
if (tej.GetTerCursorPos(CurCol) {  
    CurLine=TerGetOutInt("Line");  
    CurCol=TerGetOutInt("Col");  
}
```

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

**See Also**

[SetTerCursorPos](#)



## SetTerCursorPos

### Set the cursor position

```
boolean SetTerCursorPos( line, column, repaint)
```

```
int line; // new line position of the cursor. Set to -1 to position at  
the end of the document.
```

```
int column; // new column position of the cursor
```

```
boolean repaint; // true to refresh the window after this operation
```

**Description:** Use this function to set the new cursor position.

To specify the absolute cursor position, set the 'column' argument to -1, and specify the absolute position using the 'line' argument.

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

**See Also:**

[GetTerCursorPos](#)



## TerAbsToRowCol

### Convert the given character position to the row and column position

```
void TerAbsToRowCol( abs)
```

```
int abs; // character position (0 based) from the beginning of the  
file.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt method:**

```
int row; // To retrieve this value after a successful return from this  
method:
```

```
row=tej.TerGetOutInt("Line");
```

```
int col; // To retrieve this value after a successful return from this  
method:
```

```
col=tej.TerGetOutInt("Col");
```

**Description:** This function converts the text position given in a number of characters from the beginning of the file to the row and column position.

**Return Value:** The line and column numbers can be retrieved as following:

Example:

```
tej.TerAbsToRowCol( abs )

row=tej.TerGetOutInt( "Line" );
col=tej.TerGetOutInt( "Col" );
```

**See Also:**

[TerRowColToAbs](#)



## TerEngageCaret

**Engage the caret manually.**

```
boolean TerEngageCaret( AtCursorLoc)
```

```
boolean AtCursorLoc;           // Set to true to engage the caret at the current 'cursor'
                                location. Set to false to engage the caret at the current
                                'caret' location.
```

**Description:** The caret position indicates the text insertion point, whereas the cursor position indicates a position within currently visible text on the screen. Normally, these positions are the same. However, when the user clicks on the scrollbar, the caret can become disengaged from the cursor position. The editor will reengage the caret automatically when the user conducts any text editing operation. This function allows you to engage the caret manually. This function does not have any effect if the caret is already engaged.

**Return Value:** The function returns true when successful.



## TerGetCaretPos

**Get the current text insertion position.**

```
int TerGetCaretPos()
```

**Description:** This function returns the current text insertion position or the caret position. Please note that the editor differentiates between the text insertion position and the current cursor position as returned by the GetTerCursorPos function. When the user scrolls the text, the caret position (where the next text input will be inserted) remains the same. However, the cursor position changes in such a way that the cursor position is



REL\_WINDOW: When specifying the x/y values relative to the client area of the edit control.

REL\_TEXT\_BOX: When specifying the x/y values relative to the top of the text box.

```
int x; // The x position of the pixel.
```

```
int y; // The y position of the pixel.
```

```
// To retrieve the absolute text position in the pLine parameter, set the pCol parameter to null or set the column number to -1.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt method:**

```
int pLine; // Upon successful return from this method, the corresponding line number can be retrieved as following:  
line=tej.TerGetOutInt("Line");
```

```
int pCol; // Upon successful return from this method, the corresponding line number can be retrieved as following:  
col=tej.TerGetOutInt("Col");
```

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

Example: The example below retrieves the text position at a pixel position x=100, y=100.

```
int line;  
int col=0; // set to 0 to retrieve the position in  
// line/column format.  
if (TerPixToTextPos( tej.REL_TEXT_BOX, 100, 100, col)) {  
    line=tej.TerGetOutInt("Line");  
    col=tej.TerGetOutInt("Col");  
}
```

**See Also:**

[TerTextPosToPix](#)



**TerPosLineAtTop**

**Position the specified line at the top or middle of the window.**

boolean TerPosLineAtTop ( LineNo, WinTop)

int LineNo; // Line number to position at

boolean WinTop; // true to position the specified line at the top of the window, false to position the line at the middle of the window.

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



## TerPosBodyText

**Position the cursor at the body text outside of header or footer text.**

boolean TerPosBodyText ( section, pos, repaint)

int section; // Sequential section number for the text. Specify a number between 0 (first section) and total section -1.

This function uses sequential section numbers within the document. Please note that the sequential section numbers can be different from the actual section id for the section. You can use the TerGetSeqSect function to translate a section id into the sequential section number.

int pos; // Set to POS\_BEG to position at the first character of the body text. Set to POS\_END to position at the end of the section.

boolean repaint; // true to refresh the screen after this operation.

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

### See Also:

[TerPosHdrFtr](#)

[TerPosFrame](#)

[TerPosTable](#)

[TerGetSeqSect](#)



## TerRowColToAbs

**Convert the given line/row position to the character position.**

```
int TerRowColToAbs( row, col)
```

```
int row;                // text line number. The text line number must be  
                        // between 0 and TotalLines - 1.
```

```
int col;                // text column position. The text column position must be  
                        // between 0 and line length minus 1.
```

**Description:** This function translates the text position given in line number and column number to the character position from the beginning of the file.

**Return Value:** The function returns the text position from the beginning of the file.

**See Also:**

[TerAbsToRowCol](#)



## TerScrToTwipsX

**Translate screen x position to margin relative x position.**

```
boolean TerScrToTwipsX( ScrX)
```

```
int ScrX;                // The screen client X position (pixels) to translate.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt method:**

```
int MargX;                // The variable to receive the left margin relative position  
                        // in twips units. Upon a successful return from this method,  
                        // the MagX value can be retrieve as following:
```

```
MargX=tej.TerGetOutInt( "TwipsX" )
```

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

**See Also:**

[TerScrToTwipsY](#)



## TerScrToTwipsY

**Translate screen y position to page relative y position.**

```
boolean TerScrToTwipsY( ScrY)
```



```
int ScrY; // The screen client Y position (pixels) to translate.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt method:**

```
int PageY; // The variable to receive the position (twips) relative to the top of the page. Upon a successful return from this method, the MagX value can be retrieve as following:  
PageY=tej.TerGetOutInt("TwipsY")
```

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

**See Also:**

[TerScrToTwipsX](#)



## TerSetCaretPos

**Set the current caret position.**

```
boolean TerSetCaretPos( CaretPos)
```

```
Long CaretPos; // New caret position. The value is specified as the character position from the beginning of the document
```

**Description:** When the caret is engaged, this function simply calls the SetTerCursorPos function to set the cursor position. When the caret is disengaged from the cursor, this value updates an intejal variable. When the caret is eventually engaged, the cursor is positioned at the new caret position.

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

**See Also:**

[TerGetCaretPos](#)



## TerTextPosToPix

**Retrieve the pixel position of the text.**

```
boolean TerTextPosToPix( RelativeTo, line, col, out x, out y)
```

```
int RelativeTo; // This parameter should be set to one of the following constants:
```

REL\_SCREEN: To return the x/y values relative to the top of the screen.

REL\_WINDOW: To return the x/y values relative to the client area of the edit control.

REL\_TEXT\_BOX: To return the x/y values relative to the top of the text box.

```
int line; // The line number of the text to find position. Set this parameter to -1 to retrieve the pixel position of the text at the current caret location.
```

```
int col; // The column number of the text to find position. To specify an absolute location, set the col to -1, and specify the absolute position in the 'line' argument.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt method:**

```
int x; // The parameter to receive the x pixel position.  
Upon a successful return from this method, the MagX value can be retrieve as following:
```

```
x=tej.TerGetOutInt("X")
```

```
int y; // The parameter to receive the y pixel position.  
Upon a successful return from this method, the MagX value can be retrieve as following:
```

```
y=tej.TerGetOutInt("Y")
```

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

**Example:** The example below retrieves the pixel position at the current cursor location relative to the top of the text box.

```
int x,y,  
if (tej.TerTextPosToPix( tej.REL_TEXT_BOX, -1, -1)) {  
    x=tej.TerGetOutInt("X")  
    y=tej.TerGetOutInt("Y")  
}
```

#### See Also

[TerPixToTextPos](#)



## Table

#### In This Chapter

[TerAdustHtmlTable](#)

[TerCellBorder](#)

[TerCellBorderColor](#)

[TerCellColor](#)

[TerCellShading](#)  
[TerCellVertAlign](#)  
[TerCellRotateText](#)  
[TerCellWidth](#)  
[TerCreateCellId](#)  
[TerCreateTable](#)  
[TerDeleteCells](#)  
[TerDeleteCellText](#)  
[TerGetCellBorderColor](#)  
[TerGetCellBorderWidth](#)  
[TerGetCellInfo](#)  
[TerGetCellInfo2](#)  
[TerGetCellParam](#)  
[TerGetRowCellCount](#)  
[TerGetRowInfo](#)  
[TerGetTableId](#)  
[TerGetTableLevel](#)  
[TerGetTablePos](#)  
[TerHtmlCellWidthFlag](#)  
[TerInsertTableCol](#)  
[TerInsertTableRow](#)  
[TerIsTableSelected](#)  
[TerMarkCells](#)  
[TerPosAfterTable](#)  
[TerPosTable](#)  
[TerReformatTable](#)  
[TerRowHeight](#)  
[TerRowPosition](#)  
[TerSelectCellText](#)  
[TerSelectCol](#)  
[TerSelectRow](#)  
[TerSelectTable](#)  
[TerSetCellInfo2](#)  
[TerSetCellParam](#)  
[TerSetHdrRow](#)  
[TerSetRowKeep](#)  
[TerSetRowTextFlow](#)  
[TerSetTableColWidth](#)  
[TerSetTableId](#)



## TerAdjustHtmlTable

**Adjust the HTML table width.**

boolean TerAdjustHtmlTable()

**Description:** This function is used in conjunction with HTML add-on product. When you add/delete a column or change a column width, you can call this function to recalculate the cell width. The cell width is calculated using the html table width specification, cell contents, and the current editor window width.

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



## TerCellBorder

### Set the borders for the table cells.

boolean TerCellBorder( select, TopWidth, BotWidth, LeftWidth, RightWidth, repaint)

boolean TerCellBorder2( select, TopWidth, BotWidth, LeftWidth, RightWidth, outline, repaint)

int select; // Cell selection for this operation:

SEL\_ALL: Select the entire table

SEL\_CELLS: Select the current cell or all highlighted cells

SEL\_COLS: Select the current column or all highlighted columns

SEL\_ROWS: Select the current row or all highlighted rows

Set the 'select' parameter to 0 to invoke the user selection dialog box.

int TopWidth; Width of the top border in twips.

int BotWidth; Width of the bottom border in twips

int LeftBorder; Width of the left border in twips

int RightBorder; Width of the right border in twips

boolean outline; Set to True to draw the outline around the selected cells.

boolean repaint; true to repaint the screen after this operation

**Description:** The cursor must be positioned in a table cell before calling this function. The maximum border width should be less than the cell text margin. Any width parameter can be set to -1 to leave the current value unchanged.

**Return Value:** This function returns a true value if successful.

### See Also:

[TerPosTable](#)

[TerSetTableCoWidth](#)

[TerInsertTableRow](#)



## TerCellBorderColor

### Set the borders color for the table cells.

boolean TerCellBorderColor( select, top, bot, left, right, repaint)

boolean TerCellBorderColorHtml( select, top, bot, left, right, repaint)

int select; // Cell selection for this operation:

SEL\_ALL: Select the entire table

SEL\_CELLS: Select the current cell or all highlighted cells

SEL\_COLS: Select the current column or all highlighted columns

SEL\_ROWS: Select the current row or all highlighted rows

Set the 'select' parameter to 0 to invoke the user selection dialog box.

Color top; // Color for the top border. Set the this parameter to CLR\_ERROR to leave it unchanged.

This parameter can also be passed an html String format to specify the color, ex: "red", "#FFFFFF"

Color bot; // Color for the bottom border. Set the this parameter to CLR\_ERROR to leave it unchanged.

This parameter can also be passed an html String format to specify the color, ex: "red", "#FFFFFF"

Color leftr; // Color for the left border. Set the this parameter to CLR\_ERROR to leave it unchanged.

This parameter can also be passed an html String format to specify the color, ex: "red", "#FFFFFF"

Color right; // Color for the right border. Set the this parameter to CLR\_ERROR to leave it unchanged.

This parameter can also be passed an html String format to specify the color, ex: "red", "#FFFFFF"

```
boolean repaint;           // true to repaint the screen after this operation
```

**Description:** The cursor must be positioned in a table cell before calling this function.

**Return Value:** This function returns a true value if successful.

#### See Also:

[TerPosTable](#)  
[TerSetTableColWidth](#)  
[TerInsertTableRow](#)  
[TerCellShading](#)  
[TerCellBorder](#)



## TerCellColor

**Set the cell background color.**

```
boolean TerCellColor( select, color, repaint)
```

```
int select;           // Cell selection for this operation:
```

SEL\_ALL: Select the entire table

SEL\_CELLS: Select the current cell or all highlighted cells

SEL\_COLS: Select the current column or all highlighted columns

SEL\_ROWS: Select the current row or all highlighted rows

Set the 'select' parameter to 0 to invoke the user selection dialog box.

```
Color color;         // Cell background color
```

This parameter can also be passed as an html String format to specify the color, ex: "red", "#FFFFFF"

```
boolean repaint;     // true to repaint the screen after this operation
```

**Description:** The cursor must be positioned in a table cell before calling this function.

**Return Value:** This function returns a true value if successful.

#### See Also:

[TerPosTable](#)  
[TerCellBorder](#)  
[TerCellShading](#)  
[TerCellWidth](#)



## TerCellShading

**Set the shading percentage for the table cells.**

boolean TerCellShading( select, percent, repaint)

int select; // Cell selection for this operation:

SEL\_ALL: Select the entire table

SEL\_CELLS: Select the current cell or all highlighted cells

SEL\_COLS: Select the current column or all highlighted columns

SEL\_ROWS: Select the current row or all highlighted rows

Set the 'select' parameter to 0 to invoke the user selection dialog box

int percent; // Shading percentage (0 to 100)

boolean repaint; // true to repaint the screen after this operation

**Description:** The cursor must be positioned in a table cell before calling this function.

**Return Value:** This function returns a true value if successful.

### See Also:

[TerPosTable](#)

[TerCellBorder](#)

[TerCellColor](#)

[TerCellWidth](#)



## TerCellVertAlign

**Set the vertical alignment for the text inside a table cell.**

boolean TerCellVertAlign( select, align, repaint)

int select; // Cell selection for this operation:

SEL\_ALL: Select the entire table

SEL\_CELLS: Select the current cell or all highlighted cells

SEL\_COLS:                      Select the current column or all highlighted columns

SEL\_ROWS:                     Select the current row or all highlighted rows

Set the 'select' parameter to 0 to invoke the user selection dialog box.

int align;                     // Set this parameter to 0 to use the default 'top'  
alignment, or set it to one of the following values:

CFLAG\_VALIGN\_CTR:       Center alignment

CFLAG\_VALIGN\_BOT:       Bottom alignment

CFLAG\_VALIGN\_BASE:      Align base line of the text

boolean repaint;             // true to repaint the screen after this operation

**Description:** The cursor must be positioned in a table cell before calling this function.

**Return Value:** This function returns a true value if successful.

#### See Also:

[TerPosTable](#)

[TerCellBorder](#)

[TerCellColor](#)



## TerCellRotateText

**Set the text rotation angle within a table cell.**

boolean TerCellRotateText(select, direction, repaint)

int select;                   // Cell selection for this operation:

SEL\_ALL:                     Select the entire table

SEL\_CELLS:                  Select the current cell or all highlighted cells

SEL\_COLS:                    Select the current column or all highlighted columns

SEL\_ROWS:                    Select the current row or all highlighted rows

Set the 'select' parameter to 0 to invoke the user selection dialog box.

int direction;                // Set it to one of the following values:



TEXT_HORZ:	Horizontal text flow.
TEXT_TOP_TO_BOT:	Top-to-Bottom vertical text flow.
TEXT_BOT_TO_TOP:	Bottom-to-Top vertical text flow.

```
boolean repaint;           // TRUE to repaint the screen after this operation
```

**Description:** The cursor must be positioned in a table cell before calling this function.

**Return Value:** This function returns a True value if successful.



## TerCellWidth

**Set the cell width and cell margin.**

```
boolean TerCellWidth( select, width, margin, repaint)
```

```
int select;           // Cell selection for this operation:

SEL_ALL:             Select the entire table

SEL_CELLS:           Select the current cell or all
                    highlighted cells

SEL_COLS:            Select the current column or all
                    highlighted columns

SEL_ROWS:            Select the current row or all
                    highlighted rows
```

Set the 'select' parameter to 0 to invoke the user selection dialog box.

```
int width;           // Cell width in twips unit. Set to -1 to leave this value
                    unchanged.

int margin;         // Cell Margin in twips unit. Set to -1 to leave this value
                    unchanged.

boolean repaint;    // true to repaint the screen after this operation
```

**Description:** The cursor must be positioned in a table cell before calling this function.

**Return Value:** This function returns a true value if successful.

**See Also:**

[TerPosTable](#)

[TerCellBorder](#)

[TerCellColor](#)



## TerCreateCellId

### Create a cell id.

int TerCreateCellId( NewRow, PrevCell, RowAlign, RowPos, RowMinHeight, CellWidth, shading, LeftWidth, RightWidth, TopWidth, BotWidth, RowSpan, ColSpan, flags)

boolean NewRow;	// Set to true when creating the first cell of a new row.
int PrevCell;	// Set this value to 0 when creating the first cell id for first row of the table. Otherwise set it to the cell id of the previous cell in the table.
int RowAlign;	// LEFT,RIGHT, CENTER. (set to 0 for default).
int RowPos;	// Row position in twips (set to 0 for default).
int RowMinHeight;	// Minimum row height in twips (set to 0 for default).
int CellWidth;	// Cell width in twips.
int shading;	// Shading percentage (set to 0 for default)
int LeftWidth;	// The left border width (set to 0 for default).
int RightWidth;	// The right border width (set to 0 for default).
int TopWidth;	// The top border width (set to 0 for default).
int BotWidth;	// The bottom border width (set to 0 for default).
int RowSpan;	// Number of rows spanned by the cell (set to 1 for default).
int ColSpan;	// Number of columns spanned by the cell (set to 1 for default).
int flags;	// CFLAG_ constants defined in the tej.h file (set to 0 for default).

**Description:** This function is used to create a new cell id. This function is useful for creating tables very efficiently.

The following example creates a 3 row by 2 column table:

```
int PrevCell,CellId;
boolean NewRow;
String str;

PrevCell=CellId=0;
for (int row=0;row<3;row++) {
    for (int col=0;col<2;col++) {
        if (col==0) NewRow=true;
        else NewRow=false;
        CellId=tej.TerCreateCellId(NewRow,PrevCell,0,0,0,
                                   2000,0,0,0,0,0,1,1,0);
        str = "cell text" + tej.CELL_CHAR.ToString(); // append
                                                       cell delimiter to the cell text
        tej.TerAppendTextEx(str,-1,-1,CellId,-1,false);
        PrevCell=CellId;
    }
    str = tej.ROW_CHAR.ToString(); // insert a row delimiter
    tej.TerAppendTextEx(str,-1,-1,CellId,-1,false);
}
```

**Return Value:** When successful, this function returns the id of the new cell. Otherwise it returns 0

#### See Also:

[TerCreateParald](#)  
[TerAppendTextEx](#)  
[TerSetCellInfo2](#)



## TerCreateTable

**Create a text table.**

**boolean TerCreateTable( row, col, repaint)**

**boolean TerCreateTable2( row, col, AutoWidth, repaint)**

int row; // number of text rows in the table.

int col; // number of text columns in the table

boolean AutoWidth; // Set to TRUE to let the table cell automatically expand as the user types text into a table cell. This parameter is applicable to only the TerCreateTable2 method.

boolean repaint; //Repaint the window after this operation

**Description:** This function is used to create a text table. The number of rows and columns are specified by the 'row' and 'col' arguments. Specify a -1 value for the 'row' if you wish to activate a user dialog for the row and column selection.

The table is inserted after the current line. After this operation the cursor is placed in the first cell of the table.

Please note that the PageMode must be turned on at the design-time for this function to work properly. If the PageMode is not turned on, the tables are displayed as a series of dashed lines.

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



## TerDeleteCells

**Delete table cells.**

boolean TerDeleteCells(select, repaint)

int select; // This flag can be set to one of the following values:

SEL\_CELLS: Delete the current cell or all selected cells.

SEL\_ROWS: Delete the current row or all selected rows.

SEL\_COLS: Delete the current column or all selected columns.

You can set the 'select' parameter to 0 to invoke a dialog box to accept this parameter from the user.

boolean repaint; // repaint the screen after this operation.

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

**See Also**

[TerDeleteCellText](#)



## TerDeleteCellText

### Delete cell contents.

boolean TerDeleteCellText(select, repaint)

```
int select; // This flag can be set to one of the following values:

SEL_CELLS: Delete the contents of the current cell
            or all selected cells.

SEL_ROWS:  Delete the contents of the current row
            or all selected rows.

SEL_COLS:  Delete the contents of the current
            column or all selected columns.
```

You can set the 'select' parameter to 0 to invoke a dialog box to accept this parameter from the user.

```
boolean repaint; // repaint the screen after this operation.
```

**Comment:** This function deletes the contents of the table cell, but the table structure is not affected.

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

### See Also

[TerDeleteBlock](#)

[TerDeleteCells](#)



## TerGetCellBorderColor

### Retrieve the border color for a specific cell id.

int TerGetCellBorderColor( CellId);

```
int CellId; // Cell id to retrieve the information for. Set to -1 to select the
            current table cell.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutColor method:**

```
Color pLeft; // The left border color.
```

Upon a successful return from this method, the color value can be retrieve as following:

```

LeftColor=tej.TerGetOutColor("LeftColor")

Color pRight; // The right border color.
Upon a successful return from this method, the color value can
be retrieve as following:
RightColor=tej.TerGetOutColor("RightColor")

Color pTop; // The top border color.
Upon a successful return from this method, the color value can
be retrieve as following:
TopColor=tej.TerGetOutColor("TopColor")

Color pBottom; // The bottom border color.
Upon a successful return from this method, the color value can
be retrieve as following:
BottomColor=tej.TerGetOutColor("BotColor")

```

**Return Value:** This function returns a true value if successful.

**See Also:**  
[TerGetCellInfo](#)



## TerGetCellBorderWidth

**Retrieve the border width about a specific cell id.**

```
int TerGetCellBorderWidth( CellId, out pLeft, out pRight, out pTop, out pBottom);
```

```
int CellId; // Cell id to retrieve the information for. Set to -1 to select the
current table cell.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt method:**

```
int pLeft; // The left border width in twips.
Upon a successful return from this method, the color value can
be retrieve as following:
LeftWidth=tej.TerGetOutInt("LeftWidth")

int pRight; // The right border width in twips.
Upon a successful return from this method, the color value can
be retrieve as following:
RightWidth=tej.TerGetOutInt("RightWidth")

```

```
int pTop; // The top border width in twips.
          Upon a successful return from this method, the color value can
          be retrieve as following:
          TopWidth=tej.TerGetOutInt("TopWidth")

int pBottom; // The bottom border width in twips.
             Upon a successful return from this method, the color value can
             be retrieve as following:
             BotWidth=tej.TerGetOutInt("BotWidth")
```

**Return Value:** This function returns a true value if successful.

**See Also:**

[TerGetCellInfo](#)



## TerGetCellInfo

**Retrieve the information about a specific cell id.**

```
int TerGetCellInfo(CellId);
```

```
int CellId; // Cell id to retrieve the information for.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt method:**

```
int RowId; // The row id for the cell.
           Upon a successful return from this method, the row-id
           value can be retrieve as following:
           RowId=tej.TerGetOutInt("RowId")
```

```
int PrevCell; // Theprevious cell in the row.
              Upon a successful return from this method, the previous-
              cell id value can be retrieve as following:
              PrevCell=tej.TerGetOutInt("PrevCell")
```

```
int NextCell; // Thenext cell in the row,
              Upon a successful return from this method, the next-cell
              id value can be retrieve as following:
              NextCell=tej.TerGetOutInt("NextCell")
```

```
int width; // Pointer to receive cell width in twips.
           Upon a successful return from this method, the cell-width
```

value can be retrieve as following:

```
width=tej.TerGetOutInt( "CellWidth" )
```

```
int border; // Pointer to receive border (true / false=0).
```

Upon a successful return from this method, the cell-width value can be retrieve as following:

```
border=tej.TerGetOutInt( "CellBorder" )
```

```
int shading; // Theshading percentage.
```

Upon a successful return from this method, the cell-width value can be retrieve as following:

```
shading=tej.TerGetOutInt( "CellShading" )
```

```
int RowSpan; // Therow span for the cell.
```

Upon a successful return from this method, the cell-width value can be retrieve as following:

```
RowSpan=tej.TerGetOutInt( "RowSpan" )
```

```
int ColSpan; // Thecolumn span for the cell.
```

Upon a successful return from this method, the cell-width value can be retrieve as following:

```
ColSpan=tej.TerGetOutInt( "ColSpan" )
```

```
int flags; // Thecell flags (CFLAG_ constants).
```

Upon a successful return from this method, the cell-width value can be retrieve as following:

```
flags=tej.TerGetOutInt( "CellFlags" )
```

**Return Value:** This function returns a true value if successful.

#### See Also:

[TerGetCellBorderWidth](#)

[TerGetRowCellCount](#)



## TerGetCellInfo2

**Retrieve the information about a specific cell id.**

```
int TerGetCellInfo2( CellId)
```

```
int CellId; // Cell id to retrieve the information for.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt and TerGetOutColor methods:**



```

Color BackColor // Upon a successful return from this method, the
                 // background color value can be retrieve as following:
                 BackColor=tej.TerGetOutColor("CellBackColor")

int margin;     // Upon a successful return from this method, the
                 // previous-cell id value can be retrieve as following:
                 margin=tej.TerGetOutInt("CellMargin")

```

**Return Value:** This function returns a true value if successful.



## TerGetCellParam

**Get additional cell parameters.**

```
int TerGetCellParam(type, id)
```

```
int type; // The parameter to retrieve:
```

CP_TEXT_ROTATION:	Return the text rotation type. Please refer to the TerCellRotateText function for a list of text rotation type constants.
CP_PARENT_CELL	The parent cell id for the requested cell.
CP_LEVEL	Nesting level for the cell.
CP_TEXT_ROTATION	Returns the following values for text rotation:  TEXT_TOP_TO_BOT: Top to bottom TEXT_BOT_TO_TOP: Bottom to top TEXT_HORZ: Horizontal
CP_WIDTH	Cell width in twips.
CP_ROW	Row id which contains the specified cell.
CP_ID	Id of the specified cell. Same as the 'id' parameter, unless the 'id' parameter is set to -1.
CP_NEXT	Id of the next cell in the current row.

CP_PREV	Id of the previous cell in the current row.
CP_ROW_WIDTH	Width (in twips) of the row which contains the specified cell.
CP_PAD_LEFT	Left padding in twips.
CP_PAD_RIGHT	Right padding in twips.
CP_PAD_TOP	Top padding in twips.
CP_PAD_BOT	Bottom padding in twips.

```
int id;           // Cell id to retrieve parameters. Set to -1 to use the cell id of the current
                  // line.

int value        // The variable to receive the requested value.
```

**Return Value:** If successful, the TerGetCellParam method returns the value of the requested parameter. A return value of CP\_ERROR indicates an error.



## TerGetRowCount

**Retrieve table row or cell count.**

```
int TerGetRowCount( GetRowCount)
```

```
boolean GetRowCount;           // Set to true to return the number of rows in the current
                                // table. Set to false to get the number of cells in the
                                // current row.
```

**Return Value:** The return value is as described above. A value of 0 indicates an error.

**See Also**

[TerGetCellInfo](#)



## TerGetRowInfo

### Retrieve information about a table row id.

boolean TerGetRowInfo( RowId)

int RowId; // The row id to extract information. Set to a negative value to specify a cell id in the row.

### Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt method:

int height; // The current row height in printer units.  
Upon a successful return from this method, the height can be retrieve as following:

```
height=tej.TerGetOutInt("RowHeight")
```

int MinHeight; // The minimum height specification for the row. This value is a negative number to indicate exact row height in twips, or a positive number to indicate the minimum row height in twips, or zero to indicate auto row height.

Upon a successful return from this method, the minimum height can be retrieve as following:

```
MinHeight=tej.TerGetOutInt("RowMinHeight")
```

int FixWidth; // The width specification for the table. This value is negative number to indicate the row with as percentage of the current screen width. A positive value indicates the row width in twips.

Upon a successful return from this method, the fix-width value can be retrieve as following:

```
FixWidth=tej.TerGetOutInt("FixWidth")
```

int PrevRow; // The previous row id.

Upon a successful return from this method, the previous row value can be retrieve as following:

```
height=tej.TerGetOutInt("RowHeight")
```

int NextRow; // The next row id.

Upon a successful return from this method, the row-id value can be retrieve as following:

```
height=tej.TerGetOutInt("RowHeight")
```

int indent; // The row indent in twips.

Upon a successful return from this method, the indentation value can be retrieve as following:

```
indent=tej.TerGetOutInt("RowIndent")
```

int flags; // The row flags. The row flags (ROWFLAG\_?) constants are defined in the tej.h file.

ROWFLAG\_HDR      Header row

ROWFLAG\_RTL      Right-to-left cell placement row

Upon a successful return from this method, the flag value can be retrieve as following:

```
flags=tej.TerGetOutInt( "RowFlags" )
```

int border;

// The table border specification. This parameter is reserved for future.

Upon a successful return from this method, the border value can be retrieve as following:

```
border=tej.TerGetOutInt( "RowBorder" )
```

int CurWidth;

// This current table width in twips units.

Upon a successful return from this method, the current width value can be retrieve as following:

```
CurWidth=tej.TerGetOutInt( "RowCurWidth" )
```

**Return Value:** The function returns true if successful.

**See Also:**

[TerRowHeight](#)



## TerGetTableId

**Retrieve the table id.**

```
int TerGetTableId( row)
```

int row;

// A table row id within a table for which to retrieve the table id. You can set this parameter to -1 to indicate the current table

**Return Value:** This function returns 0 or a positive value for the table id. A value of -1 indicates an error.

**See Also:**

[TerSetTableId](#)



## TerGetTableLevel

### Get the nested table level.

```
int TerGetTableLevel( LineNo)
```

```
int LineNo; // The line number to get the table level. Set to a negative
            // value to specify a cell id instead of a line number.
```

**Return Value:** This function returns 0 or a positive value to indicate the table level number. It returns 0 if the line (or cell id) is at outer most table or if the line does not belong to a table.



## TerGetTablePos

### Get the current table position.

```
boolean TerGetTablePos( out TableNo, out RowNo, out ColNo)
```

```
boolean TerGetTablePos2( out TableNo, out RowNo, out ColNo, ParentCell)
```

```
int ParentCell; // The parent cell id to locate a nested table. Set to 0 for default.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt method:**

```
int TableNo; // The current table number. The tables are assigned a
            // sequential number starting with 0 from the beginning of the
            // document.
```

Upon a successful return from this method, the table number can be retrieved as following:

```
TableNo=tej.TerGetOutInt("TableNo")
```

```
int RowNo; // The current table row number. The table rows are assigned a
            // sequential number starting with 0 from the beginning of the
            // current table.
```

Upon a successful return from this method, the row number can be retrieved as following:

```
RowNo=tej.TerGetOutInt("RowNo")
```

```
int ColNo; // The current table column number. The table columns are
            // assigned a sequential number starting with 0 from the
            // beginning of the current table row.
```

Upon a successful return from this method, the column number can be retrieved as following:

```
ColNo=tej.TerGetOutInt("ColNo")
```

**Return Value:** This function returns true when successful. It returns a false value if the cursor is not positioned inside a table.

**See Also:**

[TerPosTable](#)



## TerHtmlCellWidthFlag

**Set the cell width flag when using the HTML add-on.**

```
boolean TerHtmlCellWidthFlag( select, flag, repaint)
```

```
int select;                // Cell selection for this operation:

                           SEL_ALL:                Select the entire table

                           SEL_CELLS:             Select the current cell or all
                                                    highlighted cells

                           SEL_COLS:              Select the current column or
                                                    all highlighted columns

                           SEL_ROWS:              Select the current row or all
                                                    highlighted rows
```

Set the 'select' parameter to 0 to invoke the user selection dialog box.

```
int flag;                  // Use one of the following values:

                           0:                      Best Fit

                           CFLAG_FIX_WIDTH        Current Width

                           CFLAG_FIX_WIDTH_PCT    Current width as the
                                                    percentage of the table
                                                    width.
```

```
boolean repaint;          // true to repaint the screen after this operation
```

**Description:** The cursor must be positioned in a table cell before calling this function.

**Return Value:** This function returns a true value if successful.



## TerInsertTableCol

### Insert or append a table column.

boolean TerInsertTableCol( insert, AllRows, repaint)

boolean insert; // true to insert a table column before the current column, or false to append a column to the table.

boolean AllRows; // true to insert/append a column to all table rows, false to insert/append a column to the current row only

boolean repaint; // true to repaint the screen after this operation

**Description:** The cursor must be positioned in a table cell before calling this function.

**Return Value:** This function returns a true value if successful.

#### See Also:

[TerPosTable](#)

[TerSetTableColWidth](#)

[TerInsertTableRow](#)

[TerCellBorder](#)

[TerCellShading](#)



## TerInsertTableRow

### Insert or append a table row.

boolean TerInsertTableRow( insert, repaint)

boolean insert; // true to insert a table row before the current row, or false to append a row to the table.

boolean repaint; // true to repaint the screen after this operation

**Description:** The cursor must be positioned in a table cell before calling this function.

**Return Value:** This function returns a true value if successful.

#### See Also:

[TerPosTable](#)

[TerSetTableColWidth](#)

[TerInsertTableCol](#)

[TerCellBorder](#)

[TerCellShading](#)



## TerIsTableSelected

**Check if any table row or cell is selected.**

boolean TerIsTableSelected()

**Return Value:** This function returns a true value if a table row or cell is selected.

### See Also

[TerGetSelection](#)



## TerMarkCells

**Set the cell selection flags for the selected text.**

boolean TerMarkCell(hWnd, select)

HWND hWnd; // Handle of the window to be accessed

int select; // Cell selection for this operation:

SEL\_ALL: Select the entire table

SEL\_CELLS: Select the current cell or all highlighted cells

SEL\_COLS: Select the current column or all highlighted columns

SEL\_ROWS: Select the current row or all highlighted rows

**Description:** This function can be called after selecting the table cells to set the cell selection flags (CFLAG\_SEL1 and CFLAG\_SEL2). You can then use the TerGetCellInfo function to retrieve the cell flags and then test against the CFLAG\_SEL1 and CFLAG\_SEL2 constants to check if the cell is selected:

```
TerGetCellInfo(CellId, RowId, PrevCell, NextCell, width,  
               border, shading, RowSpan, ColSpan, flags)  
if (flags and (CFLAG_SEL1 or CFLAG_SEL2)) then .. cell  
               selected.
```

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





## TerPosAfterTable

### Position after the current table.

boolean TerPosAfterTable ( OuterMost, repaint)

boolean OuterMost; // Set to true to position after the outer-most table, or set to false to position after the current nested table. Set to true for default.

boolean repaint; // true to refresh the screen after this operation.

**Description:** This function is available in the Page Mode only. This function places the cursor after the current table. The cursor must already be placed inside a table before calling this function.

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

### See Also:

[TerPosHdrFtr](#)

[TerPosTable](#)

[TerPosBodyText](#)



## TerPosTable

### Position the cursor at a table cell.

boolean TerPosTable ( TableNo, RowNo, ColNo, pos, repaint)

boolean TerPosTable2 ( TableNo, RowNo, ColNo, pos, ParentCell, repaint)

boolean TerPosTable3 ( TableId, RowNo, ColNo, pos, repaint)

int TableNo; // The table number of the table to position on. The first table in the document is considered the table number zero. You can also set the TableNo to -1 to specify the current table.

int TableId; // The TerPosTable3 function uses TableId instead of TableNo. The table id is an application assigned id for a table. A table id can be assigned using the TerSetTableId function.

int RowNo; // The table row number (zero based).

int ColNo; // The column number (zero based).

```
int pos; // Set to POS_BEG to position before the first character
of the existing cell text. Set to POS_END to position at
the end of the existing cell text.

Int ParentCell; // Parent cell id for the nested tables. Set to 0 for default.

boolean repaint; // true to refresh the screen after this operation.
```

**Description:** This function is available in the Page Mode only.

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

#### See Also:

[TerPosHdrFtr](#)  
[TerGetTablePos](#)  
[TerPosAfterTable](#)  
[TerGetTableId](#)



## TerReformatTable

**Recalculate the widths for the auto-width tables cells and repaginate the document.**

```
boolean TerReformatTable(repaint)
```

```
boolean repaint; // TRUE to repaint the document after this operation
```

**Return Value:** The function returns TRUE when successful



## TerRowHeight

**Set the minimum table row height.**

```
boolean TerRowHeight( MinHeight, AllRows, refresh)
```

```
int MinHeight; // Minimum table row height in twips.
```

```
boolean AllRows; // true to apply the given height to all the rows in the
table. false to apply the height to the current row or all
the highlighted rows.
```

```
boolean refresh; // true to repaint the screen after this operation.
```

**Return Value:** The function returns true when successful.

**See Also:**

[TerRowPosition](#)

[TerGetRowInfo](#)



## TerRowPosition

**Position a table row.**

```
boolean TerRowPosition( JustFlag, AllRows, refresh)
```

```
boolean TerRowPositionEx( JustFlag, indent, AllRows, refresh)
```

```
int JustFlag;                // The position of the table row:
                              LEFT:                Left justified
                              RIGHT_JUSTIFY:       Right justified
                              CENTER:              Centered

int indent;                  // The row indentation in twips unit. The JustFlag
                              parameter is ignored when the 'indent' parameter is non-
                              zero.

boolean AllRows;            // true to apply the given position to all the rows in the
                              table. false to apply the position to the current row or the
                              all highlighted rows.

boolean refresh;           // true to repaint the screen after this operation.
```

**Return Value:** The function returns true when successful.

**See Also:**

[TerRowHeight](#)



## TerSelectCellText

**Select entire text in the current table cell.**

```
boolean TerSelectCellText( repaint)
```

```
boolean repaint;            // true to refresh the screen after this operation.
```

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

**See Also**

[TerSelectCol](#)  
[TerSelectTable](#)



## TerSelectCol

**Select the current table column.**

```
boolean TerSelectCol( repaint)
```

```
boolean repaint;           // true to refresh the screen after this operation.
```

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

**See Also:**

[TerSelectRow](#)  
[TerSelectCellText](#)  
[TerSelectTable](#)



## TerSelectRow

**Select the current table row.**

```
boolean TerSelectRow( repaint)
```

```
boolean repaint;           // true to refresh the screen after this operation.
```

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

**See Also:**

[TerSelectCol](#)  
[TerSelectTable](#)



## TerSelectTable

**Select the current table.**

```
boolean TerSelectTable( level, repaint)
```

```
int level;                 // Table level number. Set this value to -1 to specify the  
                           // current level. Set this value to 0 to specify the outmost  
                           // table.
```

boolean repaint; // true to refresh the screen after this operation.

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

#### See Also

[TerSelectCellText](#)

[TerSelectRow](#)

[TerSelectCol](#)



## TerSetCellInfo2

**Set additional information for a cell id.**

boolean TerSetCellInfo2( CellId, BackColor, margin, ParentCell)

int CellId; // The cell id to set information.

Color BackColor; // The background color (RGB) for the cell.

int margin; // The cell margin in twips. Set to -1 to leave this value unchanged.

int ParentCell; // The parent cell id for this cell. Set to -1 to leave this value unchanged.

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

#### See Also:

[TerCreateCellId](#)



## TerSetCellParam

**Set cell parameters.**

boolean TerSetCellParam(select, type, val, repaint)

int select; // Cell selection for this operation:

SEL\_ALL: Select the entire table

SEL\_CELLS: Select the current cell or all highlighted cells

SEL\_COLS: Select the current column or all

highlighted columns

SEL\_ROWS: Select the current row or all highlighted rows

Set the 'select' parameter to negative value to specify a specific cell id. For example, to specify CellId value 2, set this parameter to -2.

int type;

Parameter type to set. Select one of the constants:

CP_PAD_LEFT	Specify left padding in twips.
CP_PAD_RIGHT	Specify right padding in twips.
CP_PAD_TOP	Specify top padding in twips.
CP_PAD_BOT	Specify bottom padding in twips.

int val;

New value for the parameter specified by the 'type' parameter.

boolean repaint;

TRUE to repaint the screen after this operation

**Return Value:** This function returns a true value if successful.



## TerSetHdrRow

**Set the header row for a table.**

boolean TerSetHdrRow( CellId, set, repaint)

int CellId; // A cell id for a cell in the row. Set to 0 to assume the current table row.

boolean set; // Set to true to turn the current table row (or selected rows) into a header row. Set to false to remove this attribute.

boolean repaint; // Set to true to repaint the screen after this operation

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

**See Also:**

[TerSetRowKeep](#)



## TerSetRowKeep

**Set/reset the flag to keep a table row in one page.**

boolean TerSetHdrRow( CellId, set, repaint)

int CellId; // A cell id for a cell in the row. Set to 0 to assume the current table row.

boolean set; // true to set this flag, or false to reset this flag.

boolean repaint; // Set to true to repaint the screen after this operation.

**Description:** The editor moves the entire table row to the next page when this flag is set for the row and the page break occurs within the row.

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

---

**See Also:**

[TerSetHdrRow](#)



## TerSetRowTextFlow

**Set the right-to-left/left-to-right text flow option for the table row.**

boolean TerSetRowTextFlow( dialog, AllRows, TextFlow, refresh)

boolean dialog; // Set to true to show the user dialog.

boolean AllRows; // Set to true to apply the changes to all the rows in the table. Set to false to apply the changes to the current row or the selected rows

int TextFlow; // The text flow constant can be one of the following:

FLOW\_LTR Left-to-right text flow

FLOW\_RTL Right-to-left text flow

FLOW\_DEF Default text flow. The flow will be

determined by the document, or section level text flow specification.

boolean refresh; // true to refresh the window after this operation.

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

**See Also:**

[TerSetDocTextFlow](#)

[TerSetSectTextFlow](#)



## TerSetTableColWidth

**Set the width of the table columns.**

boolean TerSetTableColWidth( width, repaint)

int width; // column width specified in twips.

boolean repaint; // true to repaint the screen after this operation

**Description:** This function sets the width of the table column where the cursor is positioned.

**Return Value:** This function returns a true value if successful.

**See Also:**

[TerPosTable](#)

[TerInsertTableCol](#)

[TerInsertTableRow](#)

[TerCellBorder](#)

[TerCellShading](#)



## TerSetTableId

**Set an id for a table.**

boolean TerSetTableId( row, id)

int row; // A table row id within a table for which to set the table id. You can set this parameter to -1 to indicate the current table.

int id; // The table id. Use a negative number to place the id on the current table row instead of the entire table. A value of 0 assigns the default table id.



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

**See Also:**

[TerGetTableId](#)



## Hyperlink

**In This Chapter**

[TerApplyHyperlink](#)

[TerDeleteHypertext](#)

[TerFindHlinkField](#)

[TerGetHypertextEx](#)

[TerInsertHyperlink](#)

[TerSetLinkDbClick](#)

[TerUpdateHyperlinkCode](#)

[TerUpdateHyperlinkText](#)



## TerApplyHyperlink

**Apply hyperlink field to the selected text.**

```
int TerApplyHyperlink( code, repaint)
```

```
String code; // The url or other information for the hyperlink
```

```
boolean repaint; //Repaint the window after this operation
```

**Comment:** Please note that the hyperlink cursor must be enabled using the ID\_SHOW\_HYPERLINK\_CURSOR to show the hyperlink cursor.

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

**See Also:**

[TerInsertHyperlink](#)

[TerUpdateHyperlinkText](#)

[TerUpdateHyperlinkCode](#)

[TerFindHlinkField](#)



## TerDeleteHypertext

**Delete the hypertext phrase and the associated hidden text.**

```
boolean TerDeleteHypertext( LineNo, ColNo, repaint)
```

```

int LineNo;           // Line number of the hypertext phrase. Set to -1 to use
                     // the current line number and column number.

int ColNo;           // Column number of the hypertext phrase. This
                     // parameter is not used when the 'LineNo' argument is set
                     // to -1.

boolean repaint;     // repaint the screen after this operation.

```

**Return Value:** This function returns true if a hypertext phrase is found at the given location and is deleted successfully.



## TerFindHlinkField

**Locate the next hyperlink field.**

```
boolean TerFindHlinkField( CodePart1, CodePart2, pLine, pCol)
```

```

String CodePart1;    // The part of the hyperlink code or url to search for

String CodePart2;    // Another part of the hyperlink code or url to search for

int pLine;           // The line number to start the search.
                     // Upon successful return, you can retrieve the final line
                     // number value as following:
                     //     line=tej.TerGetOutInt("Line");

int pCol;            // The column number to start the search.
                     // Upon successful return, You retrieve the final column
                     // number as following:
                     //     Col=tej.TerGetOutInt("Col");

```

**Description:** This function examines the hyperlinks in the document. A hyperlink is matched if either CodeString1 or CodeString2 is found in the hyperlink code or url.

**Return Value:** This function returns true when a hyperlink is located. On successful search, the line and column number of the located hyperlink can be retrieved as following:

```

if (tej.TerFindHlinkField("www.mysite.com", "", 0, 0) {
    line=tej.TerGetOutInt("Line");
    column=tej.TerGetOutInt("Col");
}

```

**see Also:**

[TerInsertHyperlink](#)

[TerUpdateHyperlinkCode](#)

[TerUpdateHyperlinkText](#)



## TerGetHypertextEx

**Retrieve the hypertext information at the cursor position.**

```
boolean TerGetHypertextEx(select)
```

```
boolean TerGetHypertext2( LineNo, ColNo, select)
```

```
int LineNo; // The line number to examine. Set to -1 to use the
             // current line and current column number. The
             // TerGetHypertextEx function automatically examines the
             // current text position.
```

```
int ColNo; // The text column number to examine.
```

```
boolean select; // true to select (highlight) the current hypertext code and
                // phrase.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutStr method:**

```
String text; // The text part of the current hypertext
             // Upon successful return, You retrieve the hyperlink text as
             // following:
```

```
text=tej.TerGetOutStr("Text");
```

```
String code; // The hidden code part of the current hypertext.
```

```
Upon successful return, You retrieve the hypertext code
or url as following:
```

```
code=tej.TerGetOutStr("FieldCode");
```

**Return Value:** This function returns true if hypertext is found at the current cursor location.



## TerInsertHyperlink

Insert hyperlink.

int TerInsertHyperlink( text, code, PictId, repaint)

String text; // The text phrase for the hyperlink.

The style and color for the hyperlink can be modified by changing the LinkStyle and LinkColor variables using the GetTerFields/SetTerFields function before inserting the hyperlink.

This parameter is ignored when the PictId parameter is non-zero.

String code; // The url or other information for the hyperlink. This information is not displayed on the screen.

int PictId; // The picture id for the hyperlink if this a picture link. Set this parameter to 0 to insert text type hyperlink.

boolean repaint; //Repaint the window after this operation

**Comment:** Please note that the hyperlink cursor must be enabled using the ID\_SHOW\_HYPERLINK\_CURSOR to show the hyperlink cursor.

**Return Value:** This function returns the font id or the picture id for the newly inserted hyperlink. It returns -1 to indicate an error condition.

#### See Also:

[TerApplyHyperlink](#)

[TerUpdateHyperlinkText](#)

[TerUpdateHyperlinkCode](#)

[TerFindHlinkField](#)



## TerSetLinkDbIClick

**Set mouse click type (single or double click) to invoke a link.**

boolean TerSetLinkDbIClick(DbIClick)

boolean DbIClick; //Set to TRUE to invoke hyperlink on a double-click. Set to FALSE to invoke hyperlink on a single click.

**Return Value:** This function returns the previous value of the DbIClick variable.



## TerUpdateHyperlinkCode

### Update the hyperlink url

boolean TerUpdateHyperlinkCode( code)

String code; // New code or url information for the hyperlink. Set the 'code' parameter to "" to convert the hyperlink text to normal text.

**Description:** This function is used to modify the hyperlink code or url for the hyperlink under the cursor.

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

#### See Also

[TerInsertHyperlink](#)  
[TerUpdateHyperlinkText](#)  
[TerFindHlinkField](#)



## TerUpdateHyperlinkText

### Update the hyperlink text

boolean TerUpdateHyperlinkText( text, repaint)

String text; // New text phrase for the hyperlink.

boolean repaint; // true to repaint after this operation.

**Description:** This function is used to modify the hyperlink text for the hyperlink under the cursor.

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

#### See Also

[TerInsertHyperlink](#)  
[TerUpdateHyperlinkCode](#)  
[TerFindHlinkField](#)



## Mail-merge

This chapter includes the mail-merge APIs. Please refer to the [Mail Merge Support](#) chapter for additional information.

#### In This Chapter

[TerChangeField](#)  
[TerChangeFieldPicture](#)

[TerChangeFieldRtf](#)  
[TerDeleteField](#)  
[TerGetField](#)  
[TerInsertField](#)  
[TerLocateField](#)  
[TerMergeFields](#)  
[TerSelectField](#)



## TerChangeField

### Change the value for a data field.

boolean TerChangeField( name, data, repaint)

String name; // Name of the field to modify

String data; // New data text for the field.

boolean repaint; // true to repaint the screen after this operation

**Description:** This function changes the data for all occurrence of the specified field name.

**Return Value:** This function returns a true value if successful.

### See Also:

[TerInsertField](#)  
[TerDeleteField](#)



## TerChangeFieldPicture

### Apply a picture to a field.

boolean TerChangeFieldPicture(name, PictPath, repaint)

String name; // Name of the field to modify

String PictPath // The picture file name or path to insert

boolean repaint; // TRUE to repaint the screen after this operation

**Description:** This function applies the picture to all occurrence of the specified field name.

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



## TerChangeFieldRtf

**Apply the rtf data to a field.**

```
boolean TerChangeFieldRtf(name, rtf, RtfLen, repaint)
```

```
String name; // Name of the field to modify
```

```
String rtf; // New RTF data for the field.
```

```
int RtfLen; // The character length of the RTF data.
```

```
boolean repaint; // TRUE to repaint the screen after this operation
```

**Description:** This function changes the data for all occurrence of the specified field name.

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



## TerDeleteField

**Delete the data field at the current cursor position.**

```
boolean TerDeleteField( repaint)
```

```
boolean repaint; // true to repaint the screen after this operation
```

**Return Value:** This function returns a true value if successful.

### See Also:

[TerInsertField](#)

[TerLocateField](#)

[TerChangeField](#)



## TerGetField

### Retrieve the text for a field name or field data.

```
int TerGetField( LineNo, ColNo, type)
```

```
int LineNo;           // The line number to examine. Set to -1 to use the
                      // current line and current column number.
```

```
int ColNo;            // The text column number to examine.
```

```
int type;             // Set to one of the following constants:
```

```
FIELD_NAME:         Retrieve the field name.
```

```
FIELD_DATA:         Retrieve the field data.
```

### Upon a successful return from this method, the following information can be retrieved by using TerGetOutStr method:

```
String text;         // sting to receive the text part for the field name or field
                      // data.
```

Upon successful return, You retrieve the field text (name or data) as following:

```
text=tej.TerGetOutStr("Text");
```

**Return Value:** This function returns the length of the retrieved text.

#### See Also:

[TerInsertField](#)

[TerLocateField](#)



## TerInsertField

### Insert a data field.

```
boolean TerInsertField( name, data, repaint)
```

```
String name;         // Field name
```

```
String data;         // Field data. Set to null to insert a field without field data.
```

```
boolean repaint;     // true to repaint the screen after the operation.
```

**Description:** This function inserts a field name and field data in the document. Please note that a data field is different from a mail merge field. A field inserted using this function can not be used for mail merge. Refer to 'Mail Merge Support' chapter for



information about mail merge fields.

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

**See Also:**

[TerGetField](#)

[TerLocateField](#)

[TerChangeField](#)

[TerDeleteField](#)



## TerLocateField

### Locate a data field.

boolean TerLocateField(location,name, exact, repaint)

```
int location;                // Use one of the following constants:

                                TER_FIRST:    Search from the top of the file and
                                locate the first occurrence of the field.

                                TER_LAST:     Search from the bottom of the file and
                                locate the last occurrence of the field.

                                TER_NEXT:     Find the next occurrence of the field.

                                TER_PREV:    Find the previous occurrence of the
                                field.

String name:                  // Field name to search for.

boolean exact:                // Set to true to match the field names in the document
                                exactly to the field name given by the 'name' parameter.
                                Set to false to match only the length of the name
                                parameter. Consider a document containing two fields
                                'company1' and 'company2'. If the name parameter is
                                set to 'company' and the 'exact' parameter is set to false,
                                then both the fields will be matched.

To match any field name, set the 'name' argument to "" and the 'exact' argument to false.

boolean repaint:              // Set to true to repaint the screen after this operation.
```

**Please note that the repaint operation can change the cursor position to adjust for any invisible text in the line. Therefore, the 'repaint' parameter should be set to false if your application relies on the cursor position set by this function for subsequent APIs.**

**Return Value:** This function returns a true value when successful. When successful, it positions the cursor on the field name.

**See Also:**

[TerGetField](#)  
[TerInsertField](#)  
[TerChangeField](#)  
[TerLocateFieldChar](#)



## TerMergeFields

**Replace field names with field data strings.**

boolean TerMergeFields(names,data,repaint)

String names; // This argument points to a list of field names. The field names must be separated by a '|' character. The list must be null terminated.

String data; // This argument points to a list containing data strings for the corresponding field names in the 'names' argument. The data strings must be separated by a '|' character. The list must be null terminated.

boolean repaint; //Repaint the window after this operation

**Description:** This function is used to replace the field names in the current editing window with the corresponding field data strings. Refer to the 'Mail/Merge Support' chapter on how to denote field names during the editing session.

If the document uses a field name which is not contained in the field name table, the editor sends a TER\_MERGE message to the parent window. The 'lParam' parameter for this message contains the variable to the field name string. If your application processes this message, it should return the variable to the field data string.

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

**See Also:**

[TerMergePrint](#)



## TerSelectField

**Select the data field at the current cursor position.**

boolean TerSelectField( SelectData, repaint)

```
boolean SelectData          // Set to TRUE to select field-data, set to false to select
                             field-name.

boolean repaint;           // TRUE to repaint the screen after this operation
```

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



## Picture and Embedded Controls

### In This Chapter

- [TerDeleteObject](#)
- [TerGetControlld](#)
- [TerGetImage](#)
- [TerGetPictCropping](#)
- [TerGetPictInfo](#)
- [TerGetPictOffset](#)
- [TerInsertControl](#)
- [TerInsertObjectId](#)
- [TerInsertPictureFile](#)
- [TerPastePicture](#)
- [TerPictAltInfo](#)
- [TerPictLinkName](#)
- [TerPictureFromFile](#)
- [TerPictureFromWmf](#)
- [TerSetBkPictId](#)
- [TerSetLinkPictDir](#)
- [TerSetPictCropping](#)
- [TerSetPictFrame2](#)
- [TerSetPictInfo](#)
- [TerSetPictOffset](#)
- [TerSetPictSize](#)
- [TerSetPlaceholderPict](#)
- [TerXlateControl](#)
- [TerSetWatermarkPict](#)
- [TerShrinkPictureToPage](#)
- [TerXlateControlld](#)



## TerDeleteObject

**Delete a font, picture or an ole object id.**

```
boolean TerDeleteObject( id)
```

```
int id;                                // A font id to delete. A font id might represent a font,
                                         picture or a control object. Your application must ensure
                                         that the font id is not in use in the document before using
```

this function to delete it.

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

**See Also:**

[TerInsertPictureFile](#)



## TerGetControlId

**Retrieve the control id for a picture id.**

```
int TerGetControlId( PictId)
```

```
int PictId; // Picture id to translate into control id
```

**Return Value:** This function returns the control id for the picture id. It returns -1 when unsuccessful.

**See Also:**

[TerInsertControl](#)

[TerXlateControlId](#)



## TerGetImage

**Retrieve the image object for a picture id.**

```
Image TerGetImage(pict)
```

```
int pict; // Id of the picture. Must be a valid number between 0 and TotalFonts - 1. Set to -1 to retrieve use the picture at the current cursor location.
```

**Return Value:** This function returns the requested Image object when successful. A null value indicates an error condition.



## TerGetPictCropping

**Retrieve picture cropping values.**

```
int TerGetPictCropping( pict, type)
```



```
int aux; // variable to receive the auxiliary id associated with the picture.

Upon successful return, You retrieve the auxiliary id as following:

aux=tej.TerGetOutInt( "AuxId" );
```

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

CrStyle, CrRectangle, CrAlign, CrAuxId

#### See Also:

[TerSetPictInfo](#)  
[TerPastePicture](#)  
[TerInsertPictureFile](#)  
[TerPictureFromFile](#)  
[TerGetPictOffset](#)  
[TerGetCurFont](#)  
[TerGetFontParam](#)



## TerGetPictOffset

**Retrieve the offset value for picture placement.**

```
int TerGetPictOffset( pict)
```

```
int pict; // Id of the picture. Must be a valid number between 0 and TotalFonts - 1.
```

**Return Value:** This function returns picture offset when successful. A value of -1 indicates an error condition.

#### See Also:

[TerGetPictInfo](#)  
[TerSetPictInfo](#)  
[TerSetPictOffset](#)



## TerInsertControl

**Insert another control inside the TEJ control.**

```
int TerInsertControl( ctl, ClassName, align, id, insert)
```

```
Control ctl; // Control to be embedded.
```

```
String ClassName;           // The class name for the control.

int align;                  // control alignment relative to the baseline of the text:
                             ALIGN_BOT (default), ALIGN_TOP, ALIGN_MIDDLE.

int id;                     // control id. Specify a unique id with a value above 2000.

boolean insert;             // true to insert the control into text. false to simply return
                             the object id without inserting the control into text.
```

**Description:** This function inserts a control of the specified class into the current text position.

When the 'insert' argument is true, the object is inserted at the current cursor location.

**Return Value:** This function returns a non-zero object id, if successful. Otherwise it returns zero.

#### See Also:

[TerInsertObjectId](#)  
[TerGetPictInfo](#)  
[TerSetPictInfo](#)  
[TerGetControllId](#)



## TerInsertObjectId

**Insert an object into text.**

```
boolean TerInsertObjectId( ObjectId, repaint)
```

```
int ObjectId;                // id of an existing object to insert.
```

```
boolean repaint;            // true to repaint the window after this operation
```

**Description:** This function inserts the specified object at the current cursor location.

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



## TerInsertPictureFile

**Embed or link a bitmap or a metafile from a disk file.**

```
int TerInsertPictureFile( FileName, embed, align, insert)
```

```
int TerInsertPictureFileXY( FileName, embed, align, insert, x, y)
```

```

int TerInsertPictureFileXY2( FileName, embed, align, insert, PageRelative, x2, y2)

String FileName;           // Name of the disk image file.

boolean embed;            // true to embed the picture in the document file, false to
                           // create a link to the picture file.

int align;                // picture alignment relative to the baseline of the text:
                           // ALIGN_BOT (default), ALIGN_TOP, ALIGN_MIDDLE.

boolean insert;           // true to insert the picture into text. false to simply return
                           // the picture id without inserting the picture into text.

int x;                    // Pixel X position to insert the picture. This parameter is
                           // used by the TerInsertPictureFileXY method only.

int y;                    // Pixel Y position to insert the picture. This parameter is
                           // used by the TerInsertPictureFileXY method only.

boolean PageRelative;     // Set to TRUE to interpret the values of the x2 and y2
                           // relative to the top of the page. Set to FALSE to insert a
                           // paragraph-relative picture.

int x2;                   // When PageRelative parameter is set to TRUE: specify
                           // the x position in twips unit relative to the left edge of the
                           // page.
                           //
                           // When PageRelative parameter is set to FALSE; specify
                           // the x position in twips unit relative to the left margin of the
                           // page.
                           //
                           // This parameter is used by the TerInsertPictureFileXY2
                           // method only.

int y2;                   // When PageRelative parameter is set to TRUE: specify
                           // the y position in twips unit relative to the top edge of the
                           // page.
                           //
                           // When PageRelative parameter is set to FALSE; specify
                           // the y position in twips unit relative to the current
                           // paragraph.
                           //
                           // This parameter is used by the TerInsertPictureFileXY2
                           // method only.

```

**Description:** A file selection dialog box is displayed if the FileName argument is set to null. The TerInsertPictureFile function inserts the picture at the current text position. The TerInsertPictureFileXY function inserts the picture at the pixel location given by the x,y position. The x,y pixel location is relative to the top left corner of the client area of the window. The TerInsertPictureFileXY2 method is used to insert a page or paragraph relative picture.

**Return Value:** This function returns a non-zero picture id, if successful. Otherwise it returns zero.



**See Also:**

[TerPastePicture](#)  
[TerPictureFromFile](#)  
[TerPictureFromWmf](#)  
[TerInsertObjectId](#)  
[TerSetBkPictId](#)  
[TerSetPictFrame2](#)  
[TerShrinkPictureToPage](#)



## TerPastePicture

**Paste a picture from the buffer or clipboard.**

int TerPastePicture( format, image, ParaFrameId, align, insert)

String format;	// A format String to specify the DataFormat class. This information is used to paste a specific data from clipboard. You can set this parameter to null to specify the default format.  This parameter is used only when the 'image' parameter is null.
Image image	// Image to paste. Set this parameter to null to paste the picture from the clipboard.
int ParaFrameId;	// Id of the frame to insert the picture into. Set 0 for default.
int align;	// picture alignment relative to the baseline of the text: ALIGN_BOT (default), ALIGN_TOP, ALIGN_MIDDLE.A
boolean insert;	// true to insert the picture into text. false to simply return the picture id without inserting the picture into text.

**Return Value:** This function returns a non-zero picture id, if successful. Otherwise it returns zero.

**See Also:**

[TerInsertPictureFile](#)  
[TerPictureFromFile](#)  
[TerInsertObjectId](#)  
[TerGetPictInfo](#)  
[TerSetPictInfo](#)  
[TerSetBkPictId](#)



## TerPictAltInfo

### Set or retrieve the alternate String for a picture.

boolean TerPictAltInfo( id, get, AltInfo)

int id; // picture id

boolean get; // true to retrieve the alternate string, false to set a new alternate string.

String AltInfo; // The new alternate name String for the picture when the get parameter is set to false.

If the 'get' parameter is set to 'true', the alternate picture information can be retrieved upon a successful return from this method as following:

```
if (tej.TerPictAltInfo(id,true,"")) {  
    String AltInfo=tej.TerGetOutStr("Info")  
}
```

**Comment:** The alternate String is saved using the 'alt' tag when the file is saved in the HTML format.

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



### TerPictLinkName

#### Set or retrieve the link file name for a picture.

boolean TerPictLinkName( id, get, FileName)

int id; // picture id

boolean get; // true to retrieve the file name, false to set a new name.

String FileName; // The new link file name for the picture if the 'get' parameter is set to false.

If the 'get' parameter is set to 'true', the link file name can be retrieved upon a successful return from this method as following:

```
if (tej.TerPictLinkName(id,true,"")) {  
    String FileName=tej.TerGetOutStr("Name")  
}
```

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



## TerPictureFromFile

This function has been discontinued. Please use the [TerInsertPictureFile](#) function instead.

### See Also:

[TerPastePicture](#)  
[TerInsertPictureFile](#)  
[TerInsertObjectId](#)  
[TerSetBkPictId](#)



## TerPictureFromWmf

This function has been discontinued. Please use the [TerInsertPictureFile](#) function instead.

### See Also:

[TerPastePicture](#)  
[TerInsertPictureFile](#)  
[TerInsertObjectId](#)  
[TerSetBkPictId](#)



## TerSetBkPictId

**Set the background picture.**

```
boolean TerSetBkPictId( PictId, PictFlag, repaint)
```

```
int PictId;           // Picture id
```

```
int PictFlag;        // This can be set to one of these values:
```

```
BKPICT_STRETCH:    Stretch the picture to occupy the  
                   text area
```

```
BKPICT_TILE:       Tile the picture to occupy the text  
                   area
```

Or you can set this argument to 0 to draw the picture in its original size without tiling.

```
boolean repaint:    // Repaint the screen after this operation
```

**Description:** The picture can be an ID returned by any of these functions: TerPastePicture, TerInsertPictureFile, TerPictureFromFile, TerPictureFromWmf. *The 'insert' argument for these function calls must be set to false.* To remove an existing picture background, call this function with the PictId set to 0. Call this function with the PictId set to -1 to show a dialog box to the user. This dialog box allows the user to select a bitmap or a metafile file name.

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

#### See Also:

[TerPastePicture](#)  
[TerInsertPictureFile](#)  
[TerPictureFromFile](#)  
[TerPictureFromWmf](#)  
[TerSetWatermarkPict](#)



## TerSetLinkPictDir

**Set the default directory to read linked pictures from an RTF file.**

boolean TerSetLinkPictDir(dir)

String dir; // The default directory. Set to "" to use the program directory.

**Description:** This directory is used to located the linked pictures which do not contain full path specification.

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



## TerSetPictCropping

**Set the picture cropping values.**

BOO TerSetPictCropping( pict, type, CropLeft, CropTop, CropRight, CropBot, repaint)

int pict; // Id of the picture. Must be a valid number between 0 and TotalFonts - 1.

int CropLeft; // Left cropping value in twips.

int CropTop; // Top cropping value in twips.

int CropRight; // Right cropping value in twips.

```
int CropBot;                // Bottom cropping value in twips.

boolean repaint;           // Repaint the screen after this operation.
```

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

#### See Also

[TerGetPictCropping](#)



## TerSetPictFrame2

### Set a floating frame for a picture.

```
boolean TerSetPictFrame2( pict, type, x, y, repaint)
```

```
int pict;                    // Id of the picture. Must be a valid number between 0
                             // and TotalFonts - 1.

int type;                    // Picture frame type:
                             // PFRAME_FLOAT: Free floating picture frame
                             // PFRAME_LEFT: Left aligned picture frame
                             // PFRAME_RIGHT: Right aligned picture frame
                             // PFRAME_NONE: No picture frame

int x;                       // The x location (in twips units) of the frame relative to
                             // the left edge of the page. This value is used only when
                             // frame type is set to PFRAME_FLOAT.

int y;                       // The y location (in twips units) of the frame relative to
                             // the top edge of the page. This value is used only when
                             // frame type is set to PFRAME_FLOAT.

boolean repaint;            // Repaint the screen after this operation.
```

**Description:** This function enclosed the give picture in a frame. The document text wraps around the frame.

**Return Value:** This function returns the frame id of the picture frame when successful. It returns a value of -1 to indicate an error condition.

#### See Also:

[TerGetPictInfo](#)

[TerSetPictSize](#)

[TerPastePicture](#)

[TerInsertPictureFile](#)



## TerSetPictInfo

**Set assorted information for a picture type object.**

boolean TerSetPictInfo( pict, style, align, aux)

int pict;	// Id of the picture. Must be a valid number between 0 and TotalFonts - 1
int style;	// Picture style constant
int align;	// Alignment flags: ALIGN_TOP, ALIGN_BOT, ALIGN_MIDDLE
int aux;	// Auxiliary id associated with the picture. This id is not used intejally by the editor. It is solely for the use of an application.

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

### See Also:

[TerGetPictInfo](#)  
[TerSetPictSize](#)  
[TerPastePicture](#)  
[TerInsertPictureFile](#)  
[TerPictureFromFile](#)



## TerSetPictOffset

**Set the offset from the baseline to place the picture in the text.**

boolean TerSetPictOffset( pict, offset, repaint)

int pict;	// Id of the picture. Must be a valid number between 0 and TotalFonts - 1.
int offset;	// The offset (twips) for the picture. This value is used to depress the bottom of the picture against the text baseline.
boolean repaint;	// true to repaint the screen after this operation.

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

### See Also:

[TerGetPictInfo](#)  
[TerSetPictInfo](#)  
[TerGetPictOffset](#)



## TerSetPictSize

**Set width and height for a picture type object.**

boolean TerSetPictSize( pict, width, height)

int pict;	// Id of the picture. Must be a valid number between 0 and TotalFonts - 1.
int width;	// New picture width in the screen units. Use the negative values to specify in twips units. Set to -1 to leave the picture width unchanged.
int height;	// New picture height in the screen units. Use the negative values to specify in twips units. Set to -1 to leave the picture height unchanged.

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

### See Also:

[TerGetPictInfo](#)  
[TerSetPictInfo](#)



## TerSetPlaceholderPict

**Insert a place-holder picture name or picture path.**

boolean TerSetPlaceholderPict( path)

String path;	// Name of the picture file, or path-name of the picture file to insert when a linked picture is missing during the RTF read process.
--------------	---

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



## TerXlateControl

Retrieve the picture id for an embedded component, or get the component object for a picture id.

**int TerXlateControl( ctl )**

**Component TerXlateControl( pict )**

Component ctl; // The component to translate into the picture id

int pict; // picture id (or font id)

**Return Value:** This function returns the picture id for the embedded component object, or returns the component object for the picture id. It returns -1 when component is not found, or null when the picture id is not found.



## TerSetWatermarkPict

**Set the watermark picture.**

boolean TerSetWatermarkPict( PictId, wash, repaint)

int PictId; // Picture id

boolean wash; // Show the watermark with a 'washed' look.

boolean repaint; // Repaint the screen after this operation

**Description:** The picture can be an ID returned by any of these functions: TerPastePicture, TerInsertPictureFile, TerPictureFromFile, TerPictureFromWmf. *The 'insert' argument for these function calls must be set to FALSE.* To remove an existing picture background, call this function with the PictId set to 0. Call this function with the PictId set to -1 to show a dialog box to the user. This dialog box allows the user to select a picture file name.

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

### See Also

[TerSetBkPictId](#)



## TerShrinkPictureToPage

**Shrink the picture to fit within a page, column, or table cell.**

boolean TerShrinkPictureToPage( line, PictId)



```
long line;                // The document line number where the picture is
                           // located. Set to -1 to specify the current line number.

int PictId;               // Picture id
```

**Description:** The picture can be an ID returned by any of these functions: TerPastePicture, TerInsertPictureFile, TerPictureFromFile, TerPictureFromWmf. *The 'insert' argument for these function calls must be set to FALSE.*

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

#### See Also

[TerInsertPictureFile](#)



## TerXlateControlld

Retrieve the picture id for a component id.

**int TerXlateControlld( CtlId)**

```
int CtlId;                // The component id to translate into the picture id
```

**Return Value:** This function returns the picture id for the component id. It returns -1 when unsuccessful.

#### See Also

[TerInsertControl](#)

[TerGetControlld](#)



## Page Header/Footer

#### In This Chapter

[TerCreateFirstHdrFtr](#)

[TerDeleteFirstHdrFtr](#)

[TerCreateLeftRightHdrFtr](#)

[TerDeleteHdrFtr](#)

[TerGetHdrFtrPos](#)

[TerHdrFtrExists](#)

[TerPosHdrFtr](#)



## TerCreateFirstHdrFtr

Create a first page header or footer area.

boolean TerCreateFirstHdrFtr( HdrFtr)

boolean HdrFtr; // Set to true to create a first page header area. Set to false to create a first page footer area.

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

**See Also:**

[TerDeleteFirstHdrFtr](#)



## TerDeleteFirstHdrFtr

**Delete a first page header or footer area.**

boolean TerDeleteFirstHdrFtr( HdrFtr, msg)

boolean HdrFtr; // Set to true to delete a first page header area. Set to false to delete a first page footer area.

boolean msg; // Set to false to suppress user confirmation before the deletion.

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

**See Also:**

[TerCreateFirstHdrFtr](#)



## TerCreateLeftRightHdrFtr

**Create a left or right page header or footer area.**

boolean TerCreateLeftRightHdrFtr(hWnd, HdrFtr)

int HdrFtr; // Specify one of the constants to create a header or footer:

RHDR\_CHAR: Right (odd) page header

RFTR\_CHAR: Right (odd) page footer

LHDR\_CHAR: Left (even) page header

LFTR\_CHAR: Left (even) page footer

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



## TerDeleteHdrFtr

**Delete a page header or footer area.**

**boolean TerDeleteHdrFtr( HdrFtr, msg)**

char HdrFtr; // Use one of the following constants to specify the type of the header/footer area to delete:

HDR\_CHAR: Regular header

FTR\_CHAR: Regular footer

FHDR\_CHAR: First page header

FFTR\_CHAR: First page footer

RHDR\_CHAR: Right (odd) page header

RFTR\_CHAR: Right (odd) page footer

LHDR\_CHAR: Left (even) page header

LFTR\_CHAR: Left (even) page footer

boolean msg; // Set to false to suppress user confirmation before the deletion.

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

### See Also:

[TerCreateFirstHdrFtr](#)



## TerGetHdrFtrPos

### See Also

[TerPosHdrFtr](#)

[TerHdrFtrExists](#)

**Check if a line is location in a header or footer area**

## int TerGetHdrFtrPos(line)

```
long line; // The line number to find the position. Set to -1 to find
           // the location for the current line.
```

**Return Value:** This function returns the following values to indicate the line position:

LFLAG_HDR	The line is location in a regular page header
LFLAG_FTR	The line is location in a regular page footer
LFLAG_FHDR	The line is location in first page header
LFLAG_FFTR	The line is location in first page footer
RFLAG_FHDR	The line is location in right (odd) page header
RFLAG_FFTR	The line is location in right (odd) page footer
LFLAG_FHDR	The line is location in left (even) page header
LFLAG_FFTR	The line is location in left (even) page footer
0	The line is location in page body text



## TerHdrFtrExists

### See Also

[TerGetHdrFtrPos](#)

[TerGetSeqSect](#)

### Check if headers or footers exists in the document

#### int TerHdrFtrExists(SectId)

```
int SectId; // The sequential section id of the section to search the
            // headers or footers. You can also set this parameter to
            // SECT_ALL to search the entire document, or set to
            // SECT_CUR to search the current section only.
```

**Return Value:** This function returns a flag value. The following constant bits in the flag value indicate the existence of the headers and footers. You can use the 'And' operator to check if a particular header or footer exists.

A zero return value indicates that no header or footer exists in the document.

LFLAG_HDR	Regular page header
LFLAG_FTR	Regular page footer
LFLAG_FHDR	First page header
LFLAG_FFTR	First page footer
LFLAG_RHDR	Right (odd) page header
LFLAG_RFTR	Right (odd) page footer
LFLAG_LHDR	Left (even) page header
LFLAG_LFTR	Left (even) page footer

**Example:**

```
int flag=tej.TerHdrFtrExists(tej.SECT_ALL);
if ((flag and tej.LFLAG_HDR) <> 0) then RegularPageHeader
                                found.
```



## TerPosHdrFtr

**Position the cursor at the header or footer text.**

boolean TerPosHdrFtr ( section, header, pos, repaint)

boolean TerPosHdrFtrEx(section,HdrFtr,pos,repaint)

int section; // section number for the header. Specify a number between 0 (first section) and total section -1.

This function uses sequential section numbers within the document. Please note that the sequential section numbers can be different from the actual section id for the section. You can use the TerGetSeqSect function to translate a section id into the sequential section number.

int header; // true to position at the header text or false to position at the footer text (used by the TerPosHdrFtr function only).

char HdrFtr; // Use one of the following constants to specify the type of the header/footer area to position at (used by the

TerPosHdrFtrEx function only):

HDR_CHAR:	Regular header
FTR_CHAR:	Regular footer
FHDR_CHAR:	First page header
FFTR_CHAR:	First Page footer
RHDR_CHAR:	Right (odd) page header
RFTR_CHAR:	Right (odd) page footer
LHDR_CHAR:	Left (even) page header
LFTR_CHAR:	Left (even) page footer

int pos; // Set to POS\_BEG to position before the first character of the existing header or footer text. Set to POS\_END to position at the end of the existing header or footer text.

boolean repaint; // true to refresh the screen after this operation.

**Description:** This function is available in the Page Mode only. The function toggles the header/footer edit mode before positioning the cursor.

Please note that this function automatically turns on the editing of header/footers, if not already enabled.

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

#### See Also:

[TerPosTable](#)  
[TerGetSeqSect](#)  
[TerGetHdrFtrPos](#)



## Frame and Drawing Objects

### In This Chapter

[TerConnectTextBoxes](#)  
[TerCreateParaFrameId](#)  
[TerGetDrawObjectInfo](#)  
[TerGetFrameParam](#)  
[TerGetFrameSize](#)  
[TerInsertDrawObject](#)  
[TerInsertLineObject](#)  
[TerInsertParaFrame](#)  
[TerMoveParaFrame](#)  
[TerMovePictFrame](#)

[TerPosFrame](#)  
[TerRotateFrameText](#)  
[TerSelectFrameText](#)  
[TerSetFrameMarginDist](#)  
[TerSetFrameTextDist](#)  
[TerSetNewFrameDim](#)  
[TerSetObjectAttrib](#)  
[TerSetObjectWrapStyle](#)  
[TerSetFrameYBase](#)



## TerConnectTextBoxes

**Link two text boxes.**

```
boolean TerConnectTextBoxes(FromFID, ToFID)
```

int FromFID; // Source frame id. Specify a number between 1 and total frames -1.

int ToFID; // Target frame id. Specify a number between 1 and total frames -1.

**Comment:** This method is used to link two text boxes. The text overflow from the first text box is displayed in the target text-box. You can use this method to construct a linked list of two or more than two text boxes.

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

### See Also:

[TerMoveParaFrame](#)  
[TerPosFrame](#)  
[TerSetNewFrameDim](#)  
[TerInsertParaFrame](#)



## TerCreateParaFrameId

**Create a paragraph frame id without inserting it in the document.**

```
int TerCreateParaFrameId( x, y, width, height)
```

int x; // X location of the frame rectangle specified in the Twips unit. The x location is relative to the left margin of the page. Specify -1 to assume the current cursor position for the x value.

int y; // Y position of the frame rectangle specified in the Twips unit. The Y position is relative to the beginning of the current paragraph. Specify -1 to assume the current

```

                                cursor position for the y value.

int width;                        // Initial width of the frame rectangle in the Twips unit.
                                Specify -1 to use the default value.

int height;                       // Initial height of the frame rectangle in the Twips unit.
                                Specify -1 to use the default value.

```

**Return Value:** When successful this function returns the paragraph frame id of the new paragraph frame. Otherwise it return 0.



## TerGetDrawObjectInfo

**Retrieve information about a drawing object.**

```
boolean TerGetDrawObjectInfo( FrameId)
```

```
int FrameId;                      // The frame id of the drawing object to inquire
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt or TerGetOutColor methods:**

```
int width;                        // The width (twips) of the text box or the rectangle object.
```

Upon a successful return from this method, the width can be retrieve as following:

```
width=tej.TerGetOutInt( "Width" )
```

```
int height;                       // The height (twips) of the text box or the rectangle object.
```

Upon a successful return from this method, the height can be retrieve as following:

```
Height=tej.TerGetOutInt( "Height" )
```

```
int LineWidth;                   // The line width (twips) of the line object.
```

Upon a successful return from this method, the line width can be retrieve as following:

```
LineWidth=tej.TerGetOutInt( "LineWidth" )
```

```
Color LineColor;                 // The color of the line object.
```

Upon a successful return from this method, the line color value can be retrieve as following:

```
LineColor=tej.TerGetOutColor( "LineColor" )
```

```
Color FillColor;                 // The fill color for the text box or the rectangle object.
```



Upon a successful return from this method, the fill-color value can be retrieve as following:

```
FillColor=tej.TerGetOutColor("BackColor")
```

```
int flags; // The object flags:

PARA_FRAME_TEXT_BOX: Indicates a text box

PARA_FRAME_LINE: Indicates a line object

PARA_FRAME_RECT: Indicates a rectangle object

PARA_FRAME_BOXED: The text box or the rectangle object is boxed.

PARA_FRAME_DOOTED: Indicates dotted border
```

Upon a successful return from this method, the row-id flag can be retrieve as following:

```
flags=tej.TerGetOutInt("flags")
```

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

**See Also:**

[TerInsertDrawObject](#)



## TerGetFrameParam

**Get the frame object parameters.**

```
int TerGetFrameParam(hWnd, id, type)
```

```
int id; // Frame item id to retrieve parameters.
```

```
int type; // The parameter to retrieve:
```

```
FP_TEXT_ROTATION: Return the text rotation type. Please refer to the TerRotateFrameText function for a list of text rotation type constants.
```

```
FP_WRAP_STYLE: Return the text wrap style. Please refer to the TerSetObjectWrapStyle function to a list of text wrap styles.
```

```
FP_YBASE Return the vertical base position for the frame. Please refer to the TerSetFrameYBase function for a list of
```

	base constant values.
FP_FILL_PATTERN	Fill pattej: 0=Transparent, 1=Solid
FP_TEXT_DIST	Text distance from the frame in twips.

**Return Value:** The function returns the value for the requested parameter. It returns FP\_ERROR to indicate an error condition.



## TerGetFrameSize

**Retrieve the location and size of the frame.**

boolean TerGetFrameSize( FrameId)

int FontId; // Frame id to inquire. If the frame id is 0, the function returns the values for the frame at the current cursor location.

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt method:**

int x; // the x location relative to the left edge of the page in twips units.

Upon a successful return from this method, the x value can be retrieve as following:

x=tej.TerGetOutInt("X")

int y; // the y location relative to the top edge of the page in twips units.

Upon a successful return from this method, the yd value can be retrieve as following:

y=tej.TerGetOutInt("Y")

int width; // the width of the frame in twips.

Upon a successful return from this method, the width value can be retrieve as following:

width=tej.TerGetOutInt("Width")

int height; // the height of the frame in twips.

Upon a successful return from this method, the height

value can be retrieve as following:

```
height=tej.TerGetOutInt("Height")
```

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

**See Also:**

[TerMoveParaFrame](#)



## TerInsertDrawObject

**Insert a drawing object.**

int TerInsertDrawObject( type, x, y, width, height)

int type;	// Drawing object type:
	DOB_TEXT_BOX: A box containing text
	DOB_RECT: A rectangle
	DOB_LINE: A horizontal line. To draw other than a horizontal line, please use the TerInsertLineObject function instead.
int x;	// X location of the frame rectangle specified in the Twips unit. The x location is relative to the left margin of the page. Specify -1 to assume the current cursor position for the x value.
int y;	// Y position of the frame rectangle specified in the Twips unit. The Y position is relative to the beginning of the current paragraph. Specify -1 to assume the current cursor position for the y value.
int width;	// Initial width of the frame rectangle in the Twips unit. Specify -1 to use the default value.
int height;	// Initial height of the frame rectangle in the Twips unit. Specify -1 to use the default value.

**Description:** This function creates a specified drawing object.

**Return Value:** When successful this function returns the paragraph frame id of the new object. Otherwise it returns 0.

**See Also:**

[TerSetObjectAttrib](#)

[TerSetFrameYBase](#)  
[TerPosFrame](#)  
[TerSetNewFrameDim](#)  
[TerInsertLineObject](#)  
[TerGetDrawObjectInfo](#)



## TerInsertLineObject

**Insert a line object.**

```
int TerInsertLineObject( x1, y1, x2, y2)
```

```
int x1;                // X position of the first point of the line.
```

```
int y1;                // Y position of the first point of the line.
```

```
int x2;                // X position of the second point of the line.
```

```
int y2;                // Y position of the second point of the line.
```

**Comments:** The point positions are specified in the twips units. The x locations are relative to the left margin of the page. The Y positions are relative to the beginning of the current paragraph.

**Return Value:** When successful this function returns the paragraph frame id of the new line object. Otherwise it returns 0.



## TerInsertParaFrame

**Insert a paragraph frame.**

```
int TerInsertParaFrame( x, y, width, height,boxed)
```

```
int x;                // X location of the frame rectangle specified in the Twips  
                    // unit. The x location is relative to the left margin of the  
                    // page. Specify -1 to assume the current cursor position  
                    // for the x value.
```

```
int y;                // Y position of the frame rectangle specified in the Twips  
                    // unit. The Y position is relative to the beginning of the  
                    // current paragraph. Specify -1 to assume the current  
                    // cursor position for the y value.
```

```
int width;           // Initial width of the frame rectangle in the Twips unit.
```

Specify -1 to use the default value.

int height; // Initial height of the frame rectangle in the Twips unit.  
Specify -1 to use the default value.

boolean boxed; // true to create a border around the frame

**Description:** This function creates an empty paragraph frame and positions the cursor inside the frame.

**Return Value:** When successful this function returns the paragraph frame id of the new paragraph frame. Otherwise it returns 0.

#### See Also:

[TerMoveParaFrame](#)  
[TerPosFrame](#)  
[TerSetNewFrameDim](#)  
[TerCreateParaFrameId](#)  
[TerSetFrameMarginDist](#)



## TerMoveParaFrame

**Move or resize the current paragraph frame or the drawing object.**

boolean TerMoveParaFrame( ParaFID, x, y, width, height)

int ParaFID; // Id of the frame which is being moved.

int x; // X location of the frame rectangle specified in the Twips unit. The x location is relative to the left margin of the page.

int y; // Y position of the frame rectangle specified in the Twips unit. The y position is relative to the top of the page when the page header/footer is visible. Otherwise it is relative to the top margin of the page.

int width; // New width of the frame rectangle in the Twips unit.  
Specify -1 to use the current value.

int height; // New height of the frame rectangle in the Twips unit.  
Specify -1 to use the current value.

**Description:** This function is used to move or resize an existing frame.

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

#### See Also:

[TerInsertParaFrame](#)  
[TerInsertDrawObject](#)

[TerGetFrameSize](#)  
[TerMovePictFrame](#)



## TerMovePictFrame

**Move a picture frame.**

boolean TerMovePictFrame(PictId, x, y)

int PictId;	// Id of the picture object. This id value must be between 0 and TotalFonts-1, and must correspond to a picture object.
int x;	// New X location of the picture frame specified in the Twips unit. The X location is relative to the left margin. Set to PARAM_IGNORE to let this value remain unchanged.
int y;	// New Y location of the picture frame specified in the Twips unit. The Y location is relative to the page top, top margin, or current paragraph as defined by the current frame Y alignment attribute. Set to PARAM_IGNORE to let this value remain unchanged.

**Description:** This function is used to move or resize an existing picture frame.

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

**See Also**

[TerSetPictSize](#)



## TerPosFrame

**Position the cursor in a frame or drawing object.**

boolean TerPosFrame ( FrameNo, pos, repaint)

int FrameNo;	// Frame id to position at. Specify a number between 1 and total frames -1.
int pos;	// Set to POS_BEG to position at the beginning of the frame. Set to POS_END to position at the end of the frame.
boolean repaint;	// true to refresh the screen after this operation.

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

**See Also:**

[TerPosBodyText](#)

[TerInsertParaFrame](#)

[TerInsertDrawObject](#)



## TerRotateFrameText

**Rotate the text within a frame or a text box.**

boolean TerRotateFrameText( dialog, LineNo, direction, repaint)

boolean dialog; // Set to true to show a dialog box to the user.

int LineNo; // The line number within a frame. You can also specify the frame id by specifying a negative value.

int direction: // Text flow direction. Choose one of the following values:

TEXT\_HORZ: Horizontal text flow

TEXT\_TOP\_TO\_BOT: Top-to-bottom text flow

TEXT\_BOT\_TO\_TOP: Bottom-to-top text flow

boolean repaint; // Set to true to repaint the screen after this operation.

Return Value: The function returns true when successful. Otherwise, it returns a false value.

**See Also:**

[TerSetObjectAttrib](#)

[TerGetFrameParam](#)

## TerSelectFrameText

**Select entire text in the current frame or text drawing object.**

boolean TerSelectFrameText( repaint)

boolean repaint; // TRUE to refresh the screen after this operation.

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



## TerSetFrameMarginDist

**Set the frame margin distance.**

```
boolean TerSetFrameMarginDist(dist)
```

```
int dist; // The frame margin distance in twips unit (default 1440 twips)
```

**Description:** The dist argument controls the minimum distance between a frame and the left or right margin at which the text starts flowing around the frame.

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

**See Also:**

[TerInsertParaFrame](#)



## TerSetFrameTextDist

**Set the frame to text distance.**

```
boolean TerSetFrameTextDist(ParaFID, dist)
```

```
int ParaFID; // The paragraph frame id to the set the distance.
```

```
int dist; // The frame text distance in twips unit (default 180 twips)
```

**Description:** The dist argument controls the minimum distance between a frame and the text flowing around the frame.

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



## TerSetNewFrameDim

**Set the default dimensions for a new frame.**

```
boolean TerSetNewFrameDim(x,y,width,height,PageTop)
```

```
int x; // The default x location for the new frame. Set this parameter to -1 to insert the new frame at the current cursor location.
```



```

int y; // The default y location for the new frame. Set this
parameter to -1 to insert the new frame at the current
cursor location.

int width; // The default width for the new frame. Set this
parameter to 0 to leave it unchanged.

int height; // The default height for the new frame. Set this
parameter to 0 to leave it unchanged.

boolean PageTop; // Set to true to create the frames relative to the top of
the page.

```

**Description:** The values passed by this function are used by any subsequent calls to the TerInsertParaFrame and TerInsertDrawObject functions as default values.

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

#### See Also:

[TerInsertParaFrame](#)  
[TerInsertDrawObject](#)



## TerSetObjectAttrib

### Set the drawing object attributes.

```
boolean TerSetObjectAttrib(ObjectId, LineType, LineThickness, LineColor, FillSolid,
FillColor)
```

```
boolean TerSetObjectAttribEx(ObjectId, LineType, LineThickness, LineColor, FillSolid,
FillColor, ZOrder)
```

```
int ObjectId; // id of the drawing object. Set to -1, to display the user
selection dialog box. This dialog box is displayed only if
the cursor is positioned on a drawing object.
```

```
int LineType; // Border line type:
```

```
DOB_LINE_NONE: No border
```

```
DOB_LINE_SOLID: Solid border line
```

```
DOB_LINE_DOTTED: Dotted border line
```

```
int LineThickness; // Border line thickness in twips.
```

```
Color LineColor; // Border line color.
```

This parameter can also be passed as an html color

String, such as "red", "blue", "#ff0000".

boolean FillSolid; // true to fill the background, false to leave the text box transparent. This option is available for a text box type drawing object only.

Color FillColor; // Background color for the drawing object.  
This parameter can also be passed as an html color String, such as "red", "blue", "#ff0000".

int ZOrder; // Z Order for the object. Set to -9999 to leave this value unchanged.

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

**See Also:**

[TerInsertDrawObject](#)  
[TerSetFrameYBase](#)  
[TerRotateFrameText](#)  
[TerSetObjectWrapStyle](#)



## TerSetObjectWrapStyle

**Set the text wrap style for a drawing object.**

boolean TerSetObjectWrapStyle(ObjectId, WrapStyle)

int ObjectId; // id of the drawing object. Set to -1, to specify the object at the current cursor position.

int WrapStyle; // Text wrap style:

SWRAP\_NO\_WRAP: Do not place text on the left and right of the object.

SWRAP\_AROUND: Flow the text around the object.

SWRAP\_THUR: Flow the text through the object.

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

**See Also**

[TerSetObjectAttrib](#)



## TerSetFrameYBase

**This function sets the vertical base position for a frame or a drawing object.**

```
boolean Ter setFrameYBase( FrameId, base)
```

```
int FrameId;           // id of the frame or the drawing object

int base;              // The base can be set to one of the following:

                        BASE_PAGE:    Vertical position
                        relative to the top of
                        the page.

                        BASE_MARG:    Vertical position
                        relative to the top
                        margin.

                        BASE_PARA:    Vertical position
                        relative to the top of
                        the anchor paragraph.
```

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

#### See Also:

[TerInsertDrawObject](#)

[TerSetObjectAttrib](#)



## Footnote, Endnote, Bookmark, Tag

### In This Chapter

[TerDeleteBookmark](#)

[TerGetBookmark](#)

[TerDeleteTag](#)

[TerGetTag](#)

[TerGetTagPos](#)

[TerInsertBookmark](#)

[TerInsertFootnote](#)

[TerPosTag](#)

[TerPosBookmark](#)

[TerSetTag](#)



## TerDeleteBookmark

**Delete a bookmark.**

```
boolean TerDeleteBookmark( name)
```

String name; // The name of the bookmark to delete.

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

**See Also:**

[TerInsertBookmark](#)

[TerPosBookmark](#)



## TerGetBookmark

**Retrieve a bookmark name.**

int TerGetBookmark( index)

int index; // The index (0 to Total Bookmarks -1) of the bookmark to retrieve. Set to -1 to retrieve the total number of bookmarks in the document.

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutStr method:**

String name; // The bookmark name.  
Upon a successful return from this method, the name can be retrieve as following:  
name=tej.TerGetOutStr( "Name" )

**Return Value:** This function returns the total number of bookmarks in the document.

**See Also:**

[TerInsertBookmark](#)

[TerPosBookmark](#)

[TerDeleteBookmark](#)



## TerDeleteTag

**Delete a tag at the specified text position:**

int TerDeleteTag (line, col, type, name)

long line; // The line position of the text. Set to -1 to use the current cursor position.

int col; // The column position of the text. This parameter is



```

        AuxText=tej.TerGetOutInt ("AuxText" )

int AuxInt;           // The auxiliary numeric data associated with the tag.
                       Upon a successful return from this method, the auxiliary
                       integer value can be retrieve as following:
                       RowId=tej.TerGetOutInt ("RowId" )

Object obj;          // The external object associated with the tag.
                       Upon a successful return from this method, the object
                       value can be retrieve as following:
                       obj=tej.TerGetOutObject ("Object" )

```

**Return Value:** This function returns the unique tag id at the specified location. It returns 0 if no tag is found at the location, or if an error is encountered.

#### See Also:

[TerPosTag](#)

[TerSetTag](#)



## TerGetTagPos

### Get the text position of a tag.

```
int TerGetTagPos(TagId, name, type)
```

```
boolean TerGetTagPos2(TagId, name, type)
```

```
int type;           // Tag type (used by the TerPosTagEx function only):
```

```
TERTAG_BKM:        Bookmark tag
```

```
TERTAG_USER:       Generic tag
```

```
int TagId;         // The tag id to search for. This parameter is ignored if
                       the 'name' parameter is specified.
```

```
String name;       // The tag name to search for. Please note that tag
                       names are not unique within a document.
```

**Upon a successful return from the TerGetTagPos2 method, the following information can be retrieved by using TerGetOutInt methods:**

```
int line;          // The line number where the tag is located.
```

This parameter is used by the TerGetTagPos2 method only. Upon a successful return from this method, the line value can be retrieve as following:

```

line=tej.TerGetOutInt("Line")

int col; // The column number where the tag is located.

This parameter is used by the TerGetTagPos2 method
only. Upon a successful return from this method, the
column value can be retrieve as following:

col=tej.TerGetOutInt("Col")

```

**Return Value:** The TerGetTagPos method searches the given tag to return the absolute character position of the tag. The function returns -1 if the tag is not found in the document.

The TerGetTagPos2 method searches the given tag to return the line/column position of the tag. The function returns a false value if the tag is not found in the document.



## TerInsertBookmark

### Insert a bookmark.

```
int TerInsertBookmark( line, col, name)
```

```
int line; // The text line number where to insert the bookmark. Set
to -1 to insert the bookmark at the current cursor
location.
```

```
int col; // The text column position to insert the bookmark. This
argument is ignored if the 'line' argument is set to -1.
```

```
String name; // Bookmark name. The name may consist of regular
alphabetic characters, but it may not contain spaces.
```

**Return Value:** This function returns a non-zero bookmark id, if successful. Otherwise it returns zero.

**See Also:** TerDeleteBookmark, TerGetBookmark, TerPosBookmark



## TerInsertFootnote

### Insert a footnote.

```
int TerInsertFootnote( FnMarker, FnText, style, repaint)
```

```

int TerInsertFootnote2( FnMarker, FnText, style, IsFootnote, repaint)

String FnMarker;           // footnote marker

String FnText;            // footnote text. Set this parameter to null to invoke a
                           // dialog box for the user to enter the footnote parameters.

int style;                // footnote marker style. The style can be any
                           // combination of BOLD, ULINE, ITALIC, and SUPSCR.
                           // Typically, the style is set to SUPSCR.

boolean IsFootnote;       // Set to true to insert a footnote. Set to false to insert a
                           // endnote. This argument is applicable to the
                           // TerInsertFootnote2 function only. The TerInsertFootnote
                           // function only inserts a footnote.

boolean repaint;          // true to repaint the screen after the operation.

```

**Description:** This function inserts a footnote at the current cursor location. The cursor is positioned after the footnote text upon the completion of this operation.

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



## TerPosTag

**Position the cursor at the specified tag position.**

```
boolean TerPosTag ( TagId, name, scope, repaint)
```

```
boolean TerPosTag Ex( type, TagId, name, scope, repaint)
```

```

int type;                  // Tag type (used by the TerPosTagEx function only):

                           TERTAG_BKM:           Bookmark tag

                           TERTAG_USER:          Generic tag

int TagId;                // The tag id to search for. This parameter is ignored if
                           // the 'name' parameter is specified.

String name;              // The tag name to search for. Please note that tag
                           // names are not unique within a document.

int scope;                // Search scope:

                           SCOPE_BEGIN:          Search from the beginning of

```



the document.

SCOPE\_FORWARD: Search below the cursor position.

SCOPE\_BACKWARD: Search above the cursor position.

SCOPE\_ANY: A fast method of locating an instance of the requested bookmark.

boolean repaint; // true to refresh the screen after this operation.

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

**See Also:**

[TerSetTab](#)

[TerGetTag](#)



## TerPosBookmark

**Position at a bookmark.**

boolean TerPosBookmark( name, repaint)

String name; // The name of the bookmark to position at.

boolean repaint; // Set to true to refresh the screen after this operation

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

**See Also:**

[TerInsertBookmark](#)

[TerGetBookmark](#)

[TerDeleteBookmark](#)



## TerSetTag

**Set a tag at the specified text position:**

int TerSetTag( line, col, name, AuxText, flags)

int TerSetTag( line, col, name, AuxText, obj, flags)

int line;	// The line position of the text. Set to -1 to use the current cursor position.
int col;	// The column position of the text. This parameter is ignored if the 'line' parameter is set to -1.
String name;	// The tag name string. A tag name does not need to be unique within the document.
String AuxText;	// Any auxiliary text String associated with the tag.
int AuxInt;	// Any auxiliary numeric data associated with the tag
object obj;	// Any external object associated with the tag.
int flags;	// Reserved for future use. Must be set to 0.

**Description:** Use this function to set a tag at the specified character position. If a tag already exists at the current text position, then the existing tag is updated with the new information.

The 'obj' parameter is not serialized to the rtf file.

**Return Value:** This function returns the unique tag id if successful. Otherwise it returns 0.

#### See Also:

[TerPosTag](#)

[TerGetTag](#)



## Stylesheet

### In This Chapter

[TerCancelEditStyle](#)

[TerDeleteStyle](#)

[TerEditStyle](#)

[TerGetStyleId](#)

[TerGetStyleInfo](#)

[TerGetStyleParam](#)

[TerSetStyleParam](#)



## TerCancelEditStyle

**Cancel the editing of the current style and restore its original value.**

boolean TerCancelEditStyle()

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

### See Also

[TerEditStyle](#)



## TerDeleteStyle

### Delete a stylesheet item.

boolean TerDeleteStyle( StyleId, name)

int StyleId; // Style id to delete. Please note that the default style ids 0 and 1 can not be deleted. Set this parameter to -1 to specify the name of the style item to delete.

String name; // Name of style item to delete. This parameter is ignored when StyleId is non-zero.

Any paragraph or font id referring the style id being deleted are modified to refer to the default style ids instead.

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

### See Also:

[TerEditStyle](#)



## TerEditStyle

### Create or edit a stylesheet style item.

int TerEditStyle( BeginRecording, name, new, type, repaint)

boolean BeginRecording; // true to begin recording a stylesheet item, false to end the recording of a stylesheet item.

String name; // Name of the stylesheet item. When this parameter is null, the editor shows a dialog box to the user to prompt for the parameters.

boolean new; // true if creating a new stylesheet item and false if modifying an existing item.

int type; // Stylesheet item type:

SSTYPE\_CHAR: Character style

SSTYPE\_PARA: Paragraph style

boolean repaint; // true to repaint the screen after this operation.

**Description:** This function is used to create a new stylesheet item or to modify an existing stylesheet item. The name and the type of the stylesheet item are specified by the 'name' and the 'type' arguments. Call this function with the 'BeginRecording' argument set to true to begin recording the properties for a stylesheet item. To apply the paragraph and character formatting properties to a stylesheet being recorded, use the menu, toolbar, ruler or any of the APIs (example: SetTerParaFmt) which modifies these properties. While a character stylesheet item is being recorded, only the character properties should be edited. When a paragraph stylesheet item is being recorded, both the character and the paragraph attributes can be edited.

To end the property recording of the current stylesheet item, call this function again with the 'BeginRecording' argument set to false.

**Return Value:** This function returns the new style id if successful. Otherwise, it returns -1.

#### See Also:

[TerSelectCharStyle](#)

[TerSelectParaStyle](#)

[TerGetFontStyleId](#)

[TerDeleteStyle](#)

[TerGetParaInfo](#)

[TerCancelEditStyle](#)



## TerGetStyleId

**Translate a stylesheet style name to style id.**

```
int TerGetStyleId( name)
```

String name; // Style name.

**Return Value:** The function returns the style id for the given style name when successful. Otherwise it returns -1.

#### See Also:

[TerDeleteStyle](#)

[TerGetStyleInfo](#)



## TerGetStyleInfo

**Retrieve information for a style item id.**

```
int TerGetStyleInfo( id, out name, out type)
```

```
int id; // Style item id to retrieve information.
```

**Upon a successful return from this method, the following information can be retrieved by using the TerGetOutInt and TerGetOutStr methods:**

```
String name; // The style name.
```

Upon a successful return from this method, the name can be retrieve as following:

```
name=tej.TerGetOutStr("Name")
```

```
int type; // The style type. The style type can be one of the following constants:
```

```
SSTYPE_PARA: Paragraph style
```

```
SSTYPE_CHAR: Character style
```

Upon a successful return from this method, the type value can be retrieve as following:

```
type=tej.TerGetOutInt("Type")
```

**Return Value:** The function returns total number of style items available in the document  
A return value of -1 indicates an error.

**See Also:**

[TerGetStyleId](#)



## TerGetStyleParam

**Get the style parameters.**

```
int TerGetStyleParam(id, type)
```

```
int id; // Style item id to retrieve parameters.
```

```
int type; // The parameter to retrieve:
```

```
SSINFO_CHAR_SPACE: Return the additional character spacing value in twips. The negative value indicates the compression of character spacing in twips unit.
```

```
SSINFO_CHAR_OFFSET Return the character offset (twips) from the baseline.
```

SSINFO\_NEXT

Returns next style id to be applied when the user hits Enter at the end of a line.

**Return Value:** The function returns the value for the requested parameter. It returns -1 to indicate an error condition.



## TerSetStyleParam

**Set the style parameters.**

boolean TerSetStyleParam(id, type, IntParam, TextParam, repaint)

int id; // Style item id to set parameters.

int type; // The parameter to set

SSINFO\_NAME: New style name. Specify using the TextParam argument.

SSINFO\_NEXT The style id to be applied to the new line when the user hits Enter at the end of a line. Specify using the IntParam argument.

int IntParam; The value for the numeric type parameter. This value is ignored for the String type parameters.

LPBYTE TextParam; The value for the String type parameter. This value is ignored for the numeric type parameters.

boolean repaint; True to repaint the screen after this operation.

**Return Value:** The function returns True when successful.



## Control Flags

In This Chapter

[TerSetFlags](#)

[TerSetFlags2](#)  
[TerSetFlags3](#)  
[TerSetFlags4](#)  
[TerSetFlags5](#)  
[TerSetFlags6](#)  
[TerSetFlags7](#)  
[TerSetFlags8](#)  
[TerSetFlags9](#)  
[TerSetFlags10](#)



## TerSetFlags

**Set certain flags or retrieve the values of the flags.**

```
int TerSetFlags( set, flags)
```

```
boolean set; // true to set the given flags, false to reset the given flags
```

```
int flags; // Flags (bits) to set or reset. Currently, the following flag values are available:
```

TFLAG_RESIZE_BITMAP:	Resize the bitmap when inserted into a frame.
TFLAG_METRIC:	Show the ruler and dialog measurements in metric units.
TFLAG_APPLY_PRT_ORIENT:	This flag overrides the paper orientation of the document with the paper orientation of the currently selected printer.
TFLAG_RETURN_MSG_ID :	Store the message ids so that they can be retrieved by using the TerGetLastMessage function.
TFLAG_IGNORE_PICT_LINK:	This flag instructs the save functions not to write the file name of the linked pictures to the disk file.
TFLAG_SHOW_CARET:	Show caret even in the read-only mode.
TFLAG_UNPROTECTED_DEL:	Allows for unprotected deletion of text blocks
TFLAG_NO_HOUR_GLASS:	Do not display the hour glass cursor during lengthy operations.
TFLAG_NO_CHILD_TOP:	Don't force child to the top of Z order.

TFLAG_NO_WRAP:	Turn-off word wrapping temporarily. This flag should not be used in the Page Mode.
TFLAG_EXCLUDE_HIDDEN_SEL:	Exclude hidden text from both ends of a selected block of text.
TFLAG_AUTO_VSCROLL_BAR:	Enable/disable vertical scroll bar depending upon the height of the text in the window. This option is not available in Page Mode.
TFLAG_NO_PALETTE:	This flag disables the intejal use of the color palettes for the picture display.
TFLAG_KEEP_PICT_ASPECT:	This flag maintains the picture aspect ratio when being resized using the mouse.
TFLAG_KEEP_FRAME_ASPECT:	This flag maintains the frame aspect ratio when being resized using the mouse.
TFLAG_PICT_IN_FRAME:	This flag creates a frame around the picture being dropped into the editor.
TFLAG_NO_PRINTER:	Disable the use of the printer.
TFLAG_NO_DRAG_TEXT:	Disable the drag/drop feature for text.
TFLAG_NO_EDIT_OLE:	Disable the editing of OLE objects
TFLAG_NO_EDIT_PICT:	Disable the editing of picture objects.
TFLAG_SHOW_BREAKS:	Show the break lines even in the read-only mode.
TFLAG_SELECT_FULL_HLINK:	Select complete hypertext phrase during text selection.
TFLAG_ROW_PASTE:	Paste the table data in the control as rows instead of as columns.
TFLAG_NO_AUTO_FULL_CELL_SEL:	Do not select the entire cell when the last character of the cell is highlighted.
TFLAG_SWAP_DECIMAL:	Swap decimal and comma characters in the dialog boxes.

**Return value:** This function returns the new value of all the flags. Call this function with the 'flags' parameter set to zero to retrieve flag values without modifying it.

**See Also:**



[TerSetFlags2](#)  
[TerSetFlags3](#)  
[TerSetFlags4](#)



## TerSetFlags2

**Set additional flags or retrieve the values of the flags.**

```
int TerSetFlags2( set, flags)
```

```
boolean set; // true to set the given flags, false to reset the given flags
```

```
int flags; // Flags (bits) to set or reset. Currently, the following flag values are available:
```

TFLAG2_RETAIN_BKND:	Display the text over the existing background. To be effective, this flag must be set before the TEJ window is created.
TFLAG2_USE_PAL_FOR_TEXT:	Use the currently selected palette to draw the text
TFLAG2_CAN_MERGE_PROT_TEXT:	Allow the deletion of the last character between two protected blocks of text.
TFLAG2_BKPICT_OVER_PAGE_BORDER:	When the background picture as well as the page border is displayed, this option ensures that the background picture is displayed in place of the border shading around the page.
TFLAG2_NO_CARET:	Do not display the caret even in the edit mode.
TFLAG2_SHOW_SECT_PAGE_NO:	Show the section page number on the status bar and while scrolling.
TFLAG2_NO_CURSOR_CHANGE:	Do not change the cursor shape as the mouse moves over the editor window.
TFLAG2_VERT_THUMB_TRACK:	Scroll the screen while dragging the vertical scroll bar.
TFLAG2_NO_AUTO_REPAGE:	Do not do automatic repagination when the document is edited.
TFLAG2_NO_BKP_FILE:	Do not save the original file as backup before saving the document.

TFLAG2_SELECT_FRAME_PICT:	When the frame containing a picture is clicked, select the picture instead of the frame.
TFLAG2_HIDE_PAGE_BREAK:	Don't show the soft page break lines in the Page Mode.
TFLAG2_PROTECT_FORMAT:	Protect the formatting of the protected text.
TFLAG2_NO_HIDDEN_RTF_TEXT:	Do not write hidden text to the RTF file.
TFLAG2_NO_SHADE_FIELD_TEXT:	Do not apply shading to the field text.
TFLAG2_IGNORE_TIMER:	Suspend timer activity.
TFLAG2_NO_AUTO_HDR_FTR:	Do not display header/footer automatically at file input or paste operation.
TFLAG2_WRITE_FIRST_RTF_COLOR:	Write the initial default color to the rtf color table.
TFLAG2_FULL_REPAINT:	Always fully repaint the text box.
TFLAG2_KEEP_PRINTER_OPEN:	Keep the printer driver open after the last edit window is closed. When this flag is set, your application should call the TerClosePriner function to close the printer driver manually after the last edit window is closed.
TFLAG2_ALT_PARA_SYM:	Display alternate paragraph symbol ( ). This flag should be set before creating the edit window.
TFLAG2_ALT_LINE_SYM:	Display alternate line break symbol ( ). This flag should be set before creating the edit window.
TFLAG2_NO_LINE_FITTING:	Do not attempt to shrink inter screen lines to fit within the printer defined width.
TFLAG2_NO_PRT_CANCEL_DLG:	Do not show the print cancel dialog box during printing.
TFLAG2_INDENT_FRAMES:	When indenting text, indent the selected frames as well.
TFLAG2_INDENT_TABLES:	When indenting text, indent the selected table as well.

TFLAG2_NO_ADJUST_CURSOR:	Do not adjust cursor when placed over hidden or protected or hidden text.
TFLAG2_CURSOR_BEF_HIDDEN:	When the cursor is in the middle of hidden text, move it before the hidden text. By default, the cursor is moved after the hidden text.
TFLAG2_NO_CURSOR_ON_PROTECT:	Do not allow cursor in the middle of the protected text.

**Return value:** This function returns the new value of all the flags. Call this function with the 'flags' parameter set to zero to retrieve flag values without modifying it.

#### See Also:

[TerSetFlags](#)  
[TerSetFlags3](#)  
[TerSetFlags4](#)



## TerSetFlags3

**Set additional flags or retrieve the values of the flags.**

```
int TerSetFlags3( set, flags)
```

```
boolean set; // true to set the given flags, false to reset the given flags
```

```
int flags; // Flags (bits) to set or reset. Currently, the following flag values are available:
```

TFLAG3_WRAP_SPACES:	Wrap additional spaces at the end of the line to the next line.
TFLAG3_NO_EDIT_TABLE_COL:	Disable the editing of table column width and indentation using the mouse.
TFLAG3_TABLE_STATUS_LINE:	Consider the table rows as lines for the status line display.
TFLAG3_PLAIN_TABLE_BORDER:	Display one line table border in HTML mode.
TFLAG3_GRAY_READ_ONLY:	Display the text in gray color in the read-only mode. To affect the color change during run-time, you also need to call the TerRepaint function after setting the read-only mode using the TerSetReadOnly

	function.
TFLAG3_CURSOR_IN_CELL:	Restrict the cursor to the current cell.
TFLAG3_NO_SCROLL:	Do not scroll down from the top of the document. This flag is effective only in the PageMode and only if the document does not contain framed text.
TFLAG3_NO_FULL_CELL_COPY:	Do not select the cell marker for cut/copy operations when the selection includes only one table cell.
TFLAG3_HTML_CONT_TABLE:	For HTML output, always treat two contiguous table rows as part of one table.
TFLAG3_SELECT_FIRST_FIELD:	Position at the first form-field in the document. This flag should be set before a file is read into the control.
TFLAG3_MULTIPLE_RTF_GROUPS:	Support consecutive 'rtf' groups in the RTF file.
TFLAG3_OLD_WORD_FORMAT:	Write RTF syntax for previous versions of MS Word.
TFLAG3_DATA_FIELD_INPUT:	Data field input mode. Protects the field name from replacement during text input to the data field.
TFLAG3_GET_BUF_HDR_FTR:	Retrieve the rtf header/footer when inserting within an existing document and when the existing document does not already have header/footer. Normally, the header/footer from the document being copied into another document is always ignored.
TFLAG3_NO_RTF_BKND_COLOR:	Do not read or write the document background color in the RTF file.
TFLAG3_NO_SAVE_UNDO:	Do not collect undo information.
TFLAG3_LARGE_PARA_BORDER:	Include the paragraph before and after spaces within the paragraph borders.
TFLAG3_LINE_SCROLL	Scroll up or down one line at a time when the up or down arrow is pressed.
TFLAG3_STYLES_ON_TOOLBAR	This flags creates a style item selection combobox in the toolbar (default). To

	remove the this combobox, turn-off this flag BEFORE the control is created.
TFLAG3_PRINT_BKND_PICT	Print any background picture when printing the document.
TFLAG3_CLIP_CELL_OVERFLOW	Do not print text lines which do not fit inside a fixed height table cell.
TFLAG3_EXACT_SCREEN_FONT	In page mode, create the exact screen font without matching it to the printer font.
TFLAG3_ZERO_CELL_HEIGHT	Do not display empty table rows.
TFLAG3_READ_PNG	Read PNG images from the RTF file.
TFLAG3_NO_TEXT_COLOR_ADJ	Do not adjust text color to contrast against the background color.
TFLAG3_NO_MOUSE_SEL	Disable text selection using mouse.

**See Also:**

[TerSetFlags](#)  
[TerSetFlags2](#)  
[TerSetFlags4](#)



## TerSetFlags4

**Set additional flags or retrieve the values of the flags.**

int TerSetFlags4( set, flags)

boolean set; // true to set the given flags, false to reset the given flags

int flags; // Flags (bits) to set or reset. Currently, the following flag values are available:

TFLAG4_COUNT_PCHAR_AS_CRLF:	Count a paragraph character as a cr/lf pair (2 characters) in TerAbsToRowCol and TerRowColToAbs functions.
TFLAG4_SKIP_PROT_TEXT:	Do not allow cursor anywhere on the protected text.
TFLAG4_READONLY_CONTROLS:	Enable or disable the embedded controls and form fields when calling the TerSetReadOnly function.

TFLAG4_NO_REPAGINATE:	Hold repagination.
TFLAG4_SMOOTH_SCROLL:	Scroll screen smoothly when selecting text using mouse.
TFLAG4_AUTO_SPELL:	Invoke auto spelling (SpellTime required for this feature)
TFLAG4_BINARY_RTF_PICT:	Save rtf pictures in the binary format (default is the hex format)
TFLAG4_UNDO_WINDOW_OVERFLOW:	Undo any editing task which makes the text go past the current window height. This flag is not effective in PageMode or FittedView modes.
TFLAG4_IME_UNICODE:	Convert DBCS characters to Unicode during IME input.
TFLAG4_MOD_END_MARK_FONT:	Set the font of the ending paragraph marker same as the font for the previous character.
TFLAG4_ONE_ROW_TOOLBAR:	Display only the second row of the toolbar.
TFLAG4_NO_OLE_DROP:	Disable dropping of OLE objects.
TFLAG4_ALWAYS_INOKE_OLE:	Invoke ole double-click even in read-only mode.
TFLAG4_DISABLE_DATE_UPDATE:	Do not update the date/time field value.
TFLAG4_SAVE_BMP_AS_PNG:	Write the the BMP images into the PNG format to the RTF output file.
TFLAG4_SAVE_SHAPE_WITH_DRAW_OBJECT:	Save shape and drawing object in one rtf file. Warning: Using this flag might generate an RTF file incompatible with MSWord.
TFLAG4_HTML_INPUT:	The input to the editor is assumed to be in the HTML format. HTML Add-on is called (if installed) to read the input data. The HTML Add-on <a href="#">license key</a> must also be set for this flag to be fully effective.
TFLAG4_NO_DRAG_PROT_TEXT:	Do not allows drag/drop of protected text.
TFLAG4_FULL_DRAG_PROT_TEXT:	Allow drag/drop of protected text fully. The protected text at the source location will be

deleted.

TFLAG4_NO_MERGE_TABLE:	Do not allow deletion of the paragraph marker before a table.
TFLAG4_NO_TOC_UPDATE:	Do not update table of contents
TFLAG4_NO_RESET_DC	Do not reset the printer device context
TFLAG4_NO_SHARE_BORDER:	Do not draw shared cell borders
TFLAG4_ADJ_LEFT_TABLE_COL	During table column adjust using the mouse, adjust only the left column.
TFLAG4_DONT_FIX_NEG_INDENT	Do not adjust for negative indents in an RTF file.
TFLAG4_PASTE_LAST_PARA_PROP	Paste the properties of the last paragraph when text is pasted into an empty paragraph.
TFLAG4_PRINT_WMF_AS_BMP	Print the metafiles as bitmap. This option is useful if the target device context is not able to handle the metafiles properly.

**See Also:**

[TerSetFlags](#)  
[TerSetFlags2](#)  
[TerSetFlags3](#)



## TerSetFlags5

**Set additional flags or retrieve the values of the flags.**

int TerSetFlags5( set, flags)

boolean set; // true to set the given flags, FALSE to reset the given flags

int flags; // Flags (bits) to set or reset. Currently, the following flag values are available:

TFLAG5_NO_EXT_DROP:	Disable text drop from other applications. This flag must be set before creating a control.
TFLAG5_GROUP_UNDO	Group the operations into one undo.

TFLAG5_NO_EXACT_ROW_HEIGHT	Translate 'Exact' row height to 'Minimum' row height during RTF input.
TFLAG5_NO_KB_SEL	Disable text selection using keyboard.
TFLAG5_RTL_CURSOR	Reverse the direction of the left/right cursor keys for the RTL text.
TFLAG5_WRITE_DOB	Generates the older "do" drawing object construct during RTF output.
TFLAG5_NO_OBJ_IN_STATUS_LINE	Ignore the embedded objects while display the status line number.
TFLAG5_NO_CLEAR_SPL_HIST	Do not clear the spell-checker word history.
TFLAG5_NO_NORMALIZE_FNOTE	Do not modified text selection to fully include footnote text and footnote marker.
TFLAG5_INPUT_TO_UNICODE	Convert keyboard input or rtf input for a foreign language to unicode.
TFLAG5_NO_NORMALIZE_FIELD	Do not modified text selection to fully include data field text in field enclosure tags in the rtf output.
TFLAG5_BEF_AND_AFT_HIDDEN	Allow the caret to stop both before and after the hidden text.
TFLAG5_FULL_REPAGINATE	Fully repaginate the document during file read irrespective of the size of the document.
TFLAG5_NO_SHOW_SPACE_SYM	Do not show the space symbol.
TFLAG5_OLD_HLINK	Allow the old hyperlink mode supporting the HLINK style and hidden/double-underline text.
TFLAG5_NO_DRAG_ROW_LINE	Do not show drag cursor to drag the row line.
TFLAG5_VARIABLE_PAGE_SIZE	<p>Automatically adjust the page size to contain the entire content of the window. When this flag is set, TE fires the <a href="#">PageSizeChanging</a> event (or TER_PAGE_SIZE_CHANGING message) to allow the user to override or modify the suggested page size.</p> <p>This flag is effective only in the Page Mode.</p>



TFLAG5_NO_DRAG_CELL_LINE	Do not show drag cursor to drag the cell divider line and row indentation.
TFLAG5_PRINT_PREVIEW_DLG	Use the .NET standard print-preview dialog instead of the built-in preview display.
TFLAG5_RULER_INDENT_FIXED	When apply indentation using the ruler to set of selected lines, set all lines to same indentation.
TFLAG5_TOP_ROW_TOOLBAR	Display only the top row of the toolbar. The TerRecreateToolbar function must be called after setting this flag to re-display the toolbar.
TFLAG5_NO_ADJ_FOR_TABLE	Do not adjust the text selection within table.
TFLAG5_NO_SHARE	Do not share font resources between the instances of the editor.
TFLAG5_FRAME_TEXT_ONLY	Do not write the frame structure while saving selected frame text to a buffer.
TFLAG5_PROTECT_DATA_FIELD	Make al data field read-only.



## TerSetFlags6

**Set additional flags or retrieve the values of the flags.**

```
int TerSetFlags6( set, flags)
```

```
boolean set;           // true to set the given flags, FALSE to reset the given
                       flags
```

```
int flags;             // Flags (bits) to set or reset. Currently, the following flag
                       values are available:
```

```
TFLAG6_WRITE_DEFAULT_COLOR  Write the default color to the rrf output
                             during clipboard paste operation.
```

```
TFLAG6_DONT_WRITE_PICT_PATH  Write linked pictures without full path.
```

```
TFLAG6_DONT_PROCESS_BULLET_K Do not use the Enter/Tab keys for
```

EYS	manipulating bullets.
TFLAG6_DONT_USE_SPELLTIME	Do not use SpellTime even if SpellTime is installed. This flag must be set after creating the Tej object, but BEFORE creating the Tej control window.
TFLAG6_INSERT_DROP_PICT_AS_LINK	Insert dropped picture files as linked pictures (and not as embedded pictures).
TFLAG6_LIST_TO_TEXT_IN_HTML	Convert list to text during HTML save.
TFLAG6_NO_LINK_MSG	Do not display a pop-up message when the mouse hovers over a text link.
TFLAG6_NO_TRACK_MSG	Do not display a pop-up message when the mouse hovers over a track-change text.
TFLAG6_SWAP_CR_LINE_BREAK	Generate line-break when Return is pressed, and generate Return when Shift-Return is pressed.
TFLAG6_DEL_CELL_TEXT	On deletion, delete cell content, but not cell structure. This flag is turned on by default.



## TerSetFlags7

### Set additional flags or retrieve the values of the flags.

```
int TerSetFlags7(set, flags)
```

```
boolean set; // TRUE to set the given flags, FALSE to reset the given flags
```

```
int flags; // Flags (bits) to set or reset. Currently, the following flag values are available:
```

TFLAG7_NO_RTL_FONT	Do not use right-to-left fonts.
--------------------	---------------------------------

TFLAG7_AUTO_NEW_ROW	Create a new row if the tab key is hit at the last column of the last row of a table. This flag is turned on by default.
---------------------	--

TFLAG7_NO_TRACK_CHANGE_LINE	Do not draw track-change indicator line.
TFLAG7_SHOW_SPACES_AT_PARA_END	Show ending spaces before the end of the paragraph.
TFLAG7_SHRINK_PICT_TO_PAGE	Shrink to size the picture to fit within a page, column, or a table cell.
TFLAG7_DISABLE_RULER	Disable ruler.
TFLAG7_ALWAYS_FIRE_MODIFY	Always fire the modified event. By default, the editor fires the modified event only for the first modification.
TFLAG7_WRITE_LISTTEXT	Generate listtext tag on RTF output so Crystal Report can show bullets.
TFLAG7_SET_BOX_CLIPPING	Draw the table cell and frame text within its boundaries.
TFLAG7_NO_SPELLCHECK_FIELDS	Do not spell-check the field text.
TFLAG7_NO_INTEJET	Disable Intejet access.
TFLAG7_NO_TABLE_AUTO_WIDTH	Disable automatic table column width adjustment.
TFLAG7_DONT_LOAD_CONTROLS	Don't read controls (originally inserted by the TerInsertControl method) while reading the rtf file.

**Return value:** This function returns the new value of all the flags. Call this function with the 'flags' parameter set to zero to retrieve flag values without modifying it.



## TerSetFlags8

**Set additional flags or retrieve the values of the flags.**

```
int TerSetFlags8(set, flags)
```

```
boolean set; // TRUE to set the given flags, FALSE to reset the given flags
```

int flags; // Flags (bits) to set or reset. Currently, the following flag values are available:

TFLAG8_AUTO_REFORMAT_TABLES	Automatically recalculate the width of the auto-width tables in the fitted-view mode
TFLAG8_DONT_SELECT_LAST_CHAR	Do not select the last character of the document on the select_all operation.
TFLAG8_HIDE_PAGE_BLANK_AREA	For short documents, disable the display of the non-text area by hiding the vertical scroll bar.
TFLAG8_MERGE_TABLES_ON_PASTE	Merge the pasted table to the preceding table.
TFLAG8_INSERT_ROWS_ON_PASTE	Insert the table rows on paste instead of overlaying it over the existing rows.
TFLAG8_DONT_SPELL_CHECK_HDR_FTR	Do not spell-check header/footer text.

**Return value:** This function returns the new value of all the flags. Call this function with the 'flags' parameter set to zero to retrieve flag values without modifying it.



## TerSetFlags9

**Set additional flags or retrieve the values of the flags.**

int TerSetFlags9(hWnd, set, flags)

boolean set; // TRUE to set the given flags, FALSE to reset the given flags

int flags; // Flags (bits) to set or reset. Currently, the following flag values are available:

TFLAG9_TREAT_DATA_FIELD_AS_ATOM	Select entire data-field for deletion or copying to clipboard.
TFLAG9_NO_COMMENTS	Do not show comment.
TFLAG9_ENABLE_POPUP_MENU	Enable the default right-click menu.

TFLAG9_INVOKE_HYPERLINK	Enable automatic invoking of the hyperlinks without requiring to handle the Hypertext event within your application.
TFLAG9_FIRE_FONT_CHECK	Allow an override of current font during rtf input or keyboard entry.

**Return value:** This function returns the new value of all the flags. Call this function with the 'flags' parameter set to zero to retrieve flag values without modifying it.



## TerSetFlags10

**Set additional flags or retrieve the values of the flags.**

int TerSetFlags10(set, flags)

boolean set; // TRUE to set the given flags, FALSE to reset the given flags

int flags; // Flags (bits) to set or reset. Currently, the following flag values are available:

TFLAG10_DONT_MARK_WITH_SEL_COL	Don't select table cells using the table column selection method
TFLAG10_DONT_PAD_TABLE_HEIGHT	Don't add the bottom padding to the exact height table rows
TFLAG10_MERGE_NESTED_RTF_PROPS	Merge nested rtf color tables
TFLAG10_EXTEND_AUTO_VSCROLL	Extend the auto-vertical-scroll bar feature to hide the vertical bar when not needed
TFLAG10_NO_HLINK_BOOKMARK	Do not generate bookmark for internal hyperlinks
TFLAG10_NO_LAST_PARA_MARKER	Don't read or write the last para-markte to the rtf file
TFLAG10_DONT_COMBINE_UNDOS	Don't combine undos
TFLAG10_AUTO_HSCROLL_BAR	Activate the automatic horizontal scroll bar feature
TFLAG10_DOTTED_GRID_LINES	Show the dotted table grid lines (instead of light blue lines)
TFLAG10_NO_CRLF	Don't insert cr/lf in the RTF text

TFLAG10_USE_LAST_PARA_PROPS	When last paragraph marker is deleted upon RTF read, use its property for the resulting last paragraph
TFLAG10_WRITE_RANDOM_LIST_TMPL_ID	Randomize the list template id when writing the RTF output
TFLAG10_LOCK_DATA_FIELD	Write the fldlock tag for the data fields
TFLAG10_SHRINK_RTF_IMAGES	Shrink images to fit during rtf input
TFLAG10_PREFER_PLACEHOLDER_PICT	Use placeholder pictures if available for the linked images
TFLAG10_NO_PICT_READ_MSG	do not show the message encountered during picture data import error
TFLAG10_DONT_USE_BKSP_FONT	Use the font of the character being deleted for subsequent insertions.

**Return value:** This function returns the new value of all the flags. Call this function with the 'flags' parameter set to zero to retrieve flag values without modifying it.



## List Numbering

The list mechanism allows you create very complex lists. Here we will described how to create simple lists.

The list mechanism consists of a list id and a list-or (list override id). These ids can be created using the TerEditList and TerEditListOr functions. One or more list-or ids can be created for a list id. However, it is sufficient in most cases to create just one or two list-or ids for each list id. The list and list-or id can have multiple levels. Each level can designate its own list numbering format (decimal, alpha, etc).

To apply list numbering to text, you would first create a list and its corresponding list-or id. Then you would apply the list-or id to the selected text (one or more paragraphs) using the TerSetParaList function. You also specify the list level when calling the TerSetParaList function.

When a list needs to be restarted from 1, it is simpler to create another list-id (and associated list-or ids).

The product also includes a function called TerSetBulletList which is a wrapper for the basic list functions in an easy to use method call.

### In This Chapter

[TerEditList](#)

[TerEditListLevel](#)

[TerEditListOr](#)  
[TerGetBulletInfo](#)  
[TerGetListInfo](#)  
[TerGetListLevelInfo](#)  
[TerGetListOrInfo](#)  
[TerSetDefListFormat](#)  
[TerSetListBullet](#)  
[TerSetListLevel](#)



## TerEditList

### Create or edit a list table item.

int TerEditList( NewList, ListId, PropDialog, ListName, nested, flags)

boolean NewList;	// Set to true to create a new list item. Set to false to modify an existing list item.
int ListId;	// The id of the list item to modify. This parameter is ignored when the 'NewList' parameter is set to true.
boolean PropDialog;	// Show a dialog box for the user to enter the list properties. When this parameter is set to true, the values for the remaining parameters are ignored.
String ListNme;	// Name of the list
boolean nested;	// Set to true to create a multi-level list. Set to false to create a single level list. A multi-level list can have up to 9 levels.
DWORD flags;	The following flag bits are supported currently:  LISTFLAG_RESTART_SEC: Restart the numbering at a section break.

**Description:** This function allows you to create a list table item. Typically, the following steps are needed to uselistnumbering for a paragraph. The first step is to use the TerEditList function to create a list table item. The second step is to use the TerEditListOr function to create one or more overrides for the list. Finally, you can use the TerSetParaList function with the list-override id and list-level parameters to assign list numbering for a paragraph. The list or list-override level properties can be modified using the TerEditListLevel function.

**Return Value:** When successful, this function returns a valid list id. A return value of -1 indicates an error condition.

### Example:

```

int ListId, ListOrId;
ListId= tej.TerEditList( true, 0, false, "MyFirstList",true, 0); // create a new multi-level list
tej.TerEditListLevel( true,ListId, 0,1, tej.LIST_DEC, tej.LISTAFT_TAB, "(~1~)", 0, 0); //
print the first level in (1), (2), (3)...format.
tej.TerEditListLevel( true,ListId, 1,1, tej.LIST_LWR_ALPHA, tej.LISTAFT_TAB,
"~1~.~2~", 0, 0); // print the second level in 1.a, 1.b. 1.c ... format.
ListOrId=tej.TerEditListOr( true, 0, false, ListId, false, 0); // create a list-override id for our
list
tej.SetTerCursorPos( 10,0,true); // position at line number 10
tej.TerSetParaList(false,-1, ListOrId, 0, true); // apply top level numbering to this line
tej.SetTerCursorPos( 11,0,true); // position at line number 11
tej.TerSetParaList(false,-1, ListOrId, 1, true); // apply second level numbering to this line

```

#### See Also:

[TerEditListOr](#)  
[TerEditListLevel](#)  
[TerSetParaList](#)  
[TerCreateListBullet](#)



## TerEditListLevel

### Edit the level properties for a list or a list-override item.

boolean TerEditListLevel( IsList, id, level, StartAt, NumType, CharAft, ListText, FontId, flags)

boolean IsList;	// Set to true to modify the level properties for a list item. Set to false to modify the level properties for a list-override item. A list-override item is allowed only if it has the 'OverrideLevels' flag set (please refer to the TerEditListOr function).
int id;	// The id of the list or list-override item to modify.
int level;	// A valid level number. A simple list has only one level (level number 0). A nested list has 9 levels (0 to 8).
int StartAt;	// The starting number for a level. A typical value would be 1.
int NumType;	// Use one of the following number type constants:
	LIST_DEC                      Decimal number



LIST_UPR_ROMAN	Uppercase roman letters
LIST_LWR_ROMAN	Lowercase roman letters
LIST_UPR_ALPHA	Uppercase alphabets
LIST_LWR_ALPHA	Lowercase alphabets
LIST_DEC_PAD	Padded decimal numbers
LIST_BLT	Bullet (no numbering)
LIST_NO_NUM	Hidden

int CharAft; // The character between the bullet text and the body text:

LISTAFT_TAB	Tabbed space
LISTAFT_SPACE	Single space
LISTAFT_NONE	No space

String ListText. // The text printed for the paragraph number. The level number information can be embedded in this text surrounded by a pair of '~' characters.

For example, specify the following list text to print the numbers in the n.n formats:

~1~.~2~

The editor will replace the ~1~ String by the current number value for level 1. Similarly, the editor will replace the ~2~ String by the current number value for level 2. The result might be as following:

- 1.1 Item 1
- 1.2 Item 2
- 1.3 Item 3

Similarly, a ListText of (~1~) would print the paragraph numbers as following:

- (1) Item 1
- (2) Item 2
- (3) Item 3

int FontId; // A valid font id (0 to TotalFonts-1) to print the paragraph number text.

DWORD flags; // One or more of the following flags bits can be specified:

LISTLEVEL_RESTART	Restart the paragraph number for a list-override level. This flag is valid only for a list-override level. When this flag is not specified, the numbering may be continued among multiple list-overrides for sharing the same list.
LISTLEVEL_REFORMAT	Use the font information as specified by the list-override level. This flag is valid only for a list-override level.
LISTLEVEL_LEGAL	This allows the previous upper level number to be printed in the Arabic numbering format.
LISTLEVEL_NO_RESET	Do not restart the level number when an upper level text is encountered.

**Description:** This function allows you edit the default properties for a list or list-override level.

**Return Value:** When successful, this function returns a true value.

**See Also:**

[TerEditList](#)  
[TerEditListOr](#)  
[TerSetParaList](#)  
[TerCreateListBullet](#)



## TerEditListOr

**Create or edit a list-override item.**

```
int TerEditListOr( NewListOr, ListOrId, PropDialog, ListId, OverrideLevels, flags)
```

```
boolean NewListOr; // Set to true to create a new list-override item. Set to false to modify an existing list-override item.
```

```
int ListOrId; // The id of the list-override item to modify. This parameter is ignored when the 'NewListOr' parameter is set to true.
```

boolean PropDialog;	// Show a dialog box for the user to enter the list-override properties. When this parameter is set to true, the values for the remaining parameters are ignored.
int ListId;	// The id of the list item to override.
boolean OverrideLevels;	// Set to true to create a new set of level information for the list-override. The level information includes properties such as number-format, starting-number, fonts, etc. When this parameter is false, the list-override item simply points to the corresponding list item without any modification.
DWORD flags;	// Set to Zero. This parameters is reserved for future use.

**Description:** The list-override items are used to override the list items. Multiple list overrides can be created for a list item. This allows for various fragments of a continuous list to be displayed in different formats. Please refer to the TerEditList function for further description of list numbering mechanism..

**Return Value:** When successful, this function returns a valid list-override id. A return value of -1 indicates an error condition.

**See Also:**

[TerEditList](#)  
[TerEditListLevel](#)  
[TerSetParaList](#)  
[TerCreateListBullet](#)



## TerGetBulletInfo

**Get the paragraph bullet/numbering information.**

int TerGetBulletInfo( QueryType, id)

int TerGetBulletInfo2( QueryType, id)

int TerGetBulletInfo3( QueryType, id)

int QueryType; // This parameter can be set to one of the following values:

PID\_LINE: Get the bullet information for the given line.

PID\_PARA: Get the bullet information for the given paragraph id.

PID\_BULLET: Get the bullet information for

the given bullet id.

PID\_STYLE                    Get the bullet information for the style id specified by the 'id' argument.

int id;                        // The value for this parameter depends upon the value specified for the 'QueryType' parameter. When the 'QueryType' is set to PID\_LINE, the 'id' parameter must contain a text line number. You can set the line number value to -1 to specify the current line. when the 'QueryType' is set to PID\_PARA, then the 'id' parameter must contain a paragraph id. When the 'QueryType' is set to PID\_BULLET, then the 'id' parameter must contain a valid bullet id. When the 'QueryType' is set to PID\_STYLE, the 'id' parameter must contain a stylesheet style id.

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt and related methods:**

boolean IsBullet;            // This flag is true if the paragraph has bullets. It is false if the paragraph numbering is turned on.

Upon a successful return from this method, this value can be retrieve as following:

```
IsBullet=tej.TerGetOutBool( "IsBullet" )
```

int start;                    // The starting number when setting paragraph numbering is turned on.

Upon a successful return from this method, the start value can be retrieve as following:

```
start=tej.TerGetOutInt( "ListStart" )
```

int level;                    // The level number when paragraph numbering is turned on.

Upon a successful return from this method, the level number can be retrieve as following:

```
level=tej.TerGetOutInt( "ListLevel" )
```

int symbol;                   // For a bullet list, the symbol can be one of the following:

BLT\_ROUND                    Round

BLT\_DIAMOND                 Diamond

BLT\_SQUARE                  Square

BLT\_HOLLOW\_SQUARE         Hollow square

BLT\_4\_DIAMOND      Four Diamonds

BLT\_ARROW          Arrow

BLT\_CHECK          Check

For a numbered list, the symbol can be one of the following:

NBR\_DEC             Decimal numbering

NBR\_UPR\_ALPHA      Uppercase letters

NBR\_LWR\_ALPHA      Lowercase letters

NBR\_UPR\_ROMAN      Uppercase Roman numbers

NBR\_LWR\_ROMAN      Lowercase Roman numbers

Upon a successful return from this method, the symbol value can be retrieve as following:

```
symbol=tej.TerGetOutInt("ListSymbol")
```

int ListOrId;

// The list override id if this bullet is created using the list table. A zero return value for this parameter indicates a regular bullet or a regular paragraph number.

A non-zero value for this field indicates a bullet or number created using the list-mechanism. In this case, the values returned by the 'type' and 'ListText' are not applicable. Please use the [TerGetListLevelInfo](#) to retrieve the list information for this list.

Upon a successful return from this method, the list override id can be retrieve as following:

```
ListOrId=tej.TerGetOutInt("ListOrId")
```

int flags;

// reserved for future use.

String ListText;

// The list text. Please refer to the [TerEditListLevel](#) function for the description of the parameter. This parameter is valid only when ListOrId is returned as non-zero.

Upon a successful return from this method, the list text can be retrieve as following:

```
ListText=tej.TerGetOutInt("ListText")
```

**Return Value:** This function returns the current bullet id. A value of 0 indicates that the

bullet/numbering is not turned on for the paragraph. A value of -1 indicates an error condition.

**See Also:**

[TerCreateBulletId](#)

[TerSetBulletEx](#)



## TerGetListInfo

### Retrieve information for a list id.

```
boolean TerGetListInfo( ListId)
```

```
boolean TerGetListInfo2( ListId)
```

```
int ListId; // The id of the list item. Set this parameter to -1 to retrieve the information about the list id for the current line.
```

The TerGetListOrInfo function can be used to retrieve the list id associated with a list override id.

### Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt and related methods:

```
String ListName; // The list name.
```

```
// This flag is true if the paragraph has bullets. It is false if the paragraph numbering is turned on.
```

Upon a successful return from this method, this value can be retrieve as following:

```
ListName=tej.TerGetOutStr("ListName")
```

```
int LevelCount; // The number of list levels supported by the list. A nested list supports 9 levels. A simple list supports one level.
```

```
// This flag is true if the paragraph has bullets. It is false if the paragraph numbering is turned on.
```

Upon a successful return from this method, this value can be retrieve as following:

```
LevelCount=tej.TerGetOutInt("ListLevelCount")
```

```
int flags; // The list flag (LISTFLAG_?). Please refer to the TerEditList for an available list of list flags.
```

```
// This flag is true if the paragraph has bullets. It is false if the paragraph numbering is turned on.
```

Upon a successful return from this method, this value can be retrieve as following:

```
flags=tej.TerGetOutInt("ListFlags")
```

```

int RtfId; // The unique RTF id for this list.

// This flag is true if the paragraph has bullets. It is false if the
// paragraph numbering is turned on.

Upon a successful return from this method, this value can be retrieve
as following:

    RtfId=tej.TerGetOutInt("ListRtfId")

int TmplId; // The template id for this list.

// This flag is true if the paragraph has bullets. It is false if the
// paragraph numbering is turned on.

Upon a successful return from this method, this value can be retrieve
as following:

    TmplId=tej.TerGetOutBool("TmplId")

```

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

#### See Also:

[TerEditList](#)  
[TerGetListOrInfo](#)  
[TerGetListLevelInfo](#)



## TerGetListLevelInfo

**Retrieve the level properties for a list or a list-override item.**

```
boolean TerGetListLevelInfo( IsList, id, level)
```

```

boolean IsList; // Set to true to retrieve the level properties for a list item.
                // Set to false to retrieve the level properties for a list-
                // override item. A list-override item is allowed only if it has
                // non-zero list levels (please refer to the TerGetListOrInfo
                // function).

int id; // The id of the list or list-override item to retrieve. You
        // can set this parameter to -1 to retrieve the information
        // about the list level for the current line.

int level; // A valid level number. A simple list has only one level
           // (level number 0). A nested list has 9 levels (0 to 8). This
           // parameter is ignored when the 'id' parameter is set to -1.

```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt and related methods:**

```
int StartAt; // The starting number for a level.
```

Upon a successful return from this method, this value can be retrieve as following:

```
StartAt=tej.TerGetOutInt("ListStart")
```

int NumType;

// The Number type used for the bullet. Please refer to the TerEditListLevel function for a list of number type constants.

Upon a successful return from this method, this value can be retrieve as following:

```
NumType=tej.TerGetOutInt("ListNumType")
```

int CharAft;

// The character between the bullet text and the body text. Please refer to the TerEditListLevel function for a list of values returned by this parameter.

Upon a successful return from this method, this value can be retrieve as following:

```
CharAft=tej.TerGetOutInt("ListCharAft")
```

String ListText.

// The text printed for the paragraph number. Please refer to the TerEditListLevel function for the description of this parameter.

Upon a successful return from this method, this value can be retrieve as following:

```
ListText=tej.TerGetOutStr("ListText")
```

int FontId;

// The font id to print the paragraph number text.

Upon a successful return from this method, this value can be retrieve as following:

```
FontId=tej.TerGetOutInt("ListFontId")
```

int flags;

// The list level flags. Please refer to the TerEditListLevel function for a description of the list level flags.

Upon a successful return from this method, this value can be retrieve as following:

```
flags=tej.TerGetOutInt("ListFlags")
```

**Return Value:** When successful, this function returns a true value.

**See Also:**

[TerGetListInfo](#)  
[TerGetListOrInfo](#)  
[TerEditListLevel](#)



## TerGetListOrInfo



### Retrieve information for a list override id.

boolean TerGetListOrInfo( ListOrId, out ListId, out LevelCount, out flags)

int ListOrId; // The id of the list override item. Set this parameter to -1 to retrieve the information about the list override id for the current line.  
The TerGetBulletInfo2 function can be used to retrieve the list override id associated with the current line or a paragraph id.

### Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt and related methods:

int ListId; // The list id for this list override id.  
Upon a successful return from this method, this value can be retrieve as following:  
`ListId=tej.TerGetOutInt("ListId")`

int LevelCount; // The number of list levels supported by the list override id. A non-zero value indicates that this list override id actually overrides the level information for the underlying list id.  
Upon a successful return from this method, this value can be retrieve as following:  
`LevelCount=tej.TerGetOutInt("ListLevelCount")`

int flags; // The list override flags. This parameter is not used currently and always returns a zero value.  
Upon a successful return from this method, this value can be retrieve as following:  
`flags=tej.TerGetOutInt("ListFlags")`

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

#### See Also:

[TerGetBulletInfo](#)  
[TerEditListOr](#)  
[TerGetListInfo](#)  
[TerGetListLevelInfo](#)



## TerSetDefListFormat

### Specify default list level properties to apply when creating a new list.

boolean TerSetDefListFormat( level, NumType, ListText)

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

```

int id; // The id of the list or list-override item to modify.

int level; // Level number to apply the properties (0 to 8).

int NumType; // Use one of the following number type constants:

LIST_DEC           Decimal number
LIST_UPR_ROMAN    Uppercase roman letters
LIST_LWR_ROMAN    Lowercase roman letters
LIST_UPR_ALPHA    Uppercase alphabets
LIST_LWR_ALPHA    Lowercase alphabets
LIST_DEC_PAD      Padded decimal numbers
LIST_BLT          Bullet (no numbering)
LIST_NO_NUM       Hidden

```

```

String ListText. // The text printed for the paragraph number. The level
                 // number information can be embedded in this text
                 // surrounded by a pair of '~' characters.
                 //
                 // For example, specify the following list text to print the
                 // numbers in the n.n formats:
                 //
                 // ~1~.~2~
                 //
                 // The editor will replace the ~1~ String by the current
                 // number value for level 1. Similarly, the editor will replace
                 // the ~2~ String by the current number value for level 2.
                 // The result might as following:
                 //
                 // 1.1 Item 1
                 // 1.2 Item 2
                 // 1.3 Item 3
                 //
                 // Similarly, a ListText of (~1~) would print the paragraph
                 // numbers as following:
                 //
                 // (1) Item 1
                 // (2) Item 2
                 // (3) Item 3

```

**Description:** The properties specified using this method is applicable only to a new list subsequently created using any of the list creation methods.

**Return Value:** When successful, this function returns a TRUE value.



## TerSetListBullet

### Set the paragraph bullet/numbering property.

```
boolean TerSetListBullet(set, NumType, level, start, TextBef, TextAft, repaint)
```

```
boolean TerSetListBullet2(set, NumType, level, start, TextBef, TextAft, ListText, repaint)
```

```
boolean set; // TRUE to set the paragraph bullet/numbering or FALSE to remove it. The 'start', 'level', and 'type' parameters are ignored when the 'set' parameter is FALSE.
```

```
int NumType // This parameter allows you to apply a bullet or a numbered list. Please refer to the TerEditListLevel function for the list of available constants.
```

```
int start; // The starting number when setting paragraph numbering. Set to 1 for default.
```

```
int level; // list level (0 to 8). The default value for this parameter is 0.
```

```
String TextBef; // Text before the paragraph number. Set to "" for default. This parameter is ignored when the ListText parameter is non-blank.
```

```
String TextAft; // Text After the paragraph number. Set to "" for default. This parameter is ignored when the ListText parameter is non-blank.
```

```
String ListText; // list-text for the bullet/number. Please refer to the TerEditListLevel function for the description of the parameter.
```

```
boolean repaint; // Repaint the screen after this operation
```

### Description:

The TerSetListBullet method is a wrapper to the basic list mechanism functions: TerEditList, TerEditListOr, TerEditListLevel and TerSetParaList. This method provides most used features of the basic list functions. However, you might like to use the basic list function if you need further control of the lists. Please refer to the [ListNumbering](#) chapter for a discussion on using the basic list functions.

### Examples:

**Create a decimal numbered list of level 0 where the numbering prints as:**

**(1), (2), (3), Etc**

```
tej.TerSetListBullet2(true,tej.LIST_DEC,0,1,"",""("~1~)",true);
```

Create a decimal numbered list of level 0 where the numbering prints as:

1., 2., 3., Etc.

```
tej.TerSetListBullet2(true,tej.LIST_DEC,0,1,"",""~1~.",true);
```

Create a decimal numbered list of level 1 where the numbering prints as:

1.1, 1.2, 1.3, etc.

```
tej.TerSetListBullet2(true,tej.LIST_DEC,1,1,"",""~1~.~2~",true);
```

Create a standard bullet list of level 1 with round bullet:

```
tej.TerSetListBullet2(true,tej.LIST_BLT,1,1,"",""~",true);
```

Create a square bullet list of level 1 :

```
tej.TerSetListBullet2(true,tej.LIST_BLT,1,1,"","",new string((char)61607,1),true);
```

// unicode 61607 shows square

Create a arrow bullet list of level 1:

```
tej.TerSetListBullet2(true,tej.LIST_BLT,1,1,"","",new string((char)61656,1),true);
```

// unicode 61656 shows arrow symbol

Examples of other unicode bullets:

unicode 61558 : 4-diamond shaped bullet

unicode 61692: Check mark bullet

You can also any character value between 33 and 255 in the Wingdings character set to display a bullet.

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

#### See Also

[TerSetListLevel](#)



## TerSetListLevel

**Set the level for a list.**

```
boolean TerSetListLevel(level, increment, repaint)
```

```
int level;
```

```
// list level (0 to 8). The outmost level is 0. Set this  
parameter to -1 to increase or decrease the level using
```

the 'increment' parameter.

int increment

// Specify the number to increase or decrease the current level. For example, a value of 1 will increase the level by one. A value of -1 will decrease the level by one.

This parameter is used only when the 'level' parameter is set to -1.

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

#### See Also

[TerSetListBullet](#)



## Toolbar

#### In This Chapter

[TerEditTooltip](#)

[TerAddToolbarIcon](#)

[TerHideToolbarIcon](#)

[TerRecreateToolbar](#)

[TerSetToolbarComboWidth](#)

[TerUpdateToobar](#)



## TerEditTooltip

**Specify a new tooltip text.**

int TerEditTooltip( id, tooltip)

int id;

// The toolbar icon id. The following is a list of toolbar icon ids:

TLB\_LINE                      Vertical line in the toolbar. Tooltip is not applicable to this id.

TLB\_TYPEFACE                 Font typeface combobox

TLB\_POINTSIZE                Font pointsize combobox

TLB\_BOLD                      Bold style

TLB\_ITALIC                    Italic style

TLB\_ULINE                     Underline style

TLB_ALIGN_LEFT	Align left
TLB_ALIGN_RIGHT	Align right
TLB_ALIGN_CENTER	Paragraph centering
TLB_ALIGN_JUSTIFY	Align both
TLB_INC_INDENT	Increase paragraph indentation
TLB_DEC_INDENT	Decrease paragraph indentation
TLB_STYLE	Stylesheet item combobox
TLB_ZOOM	Zoom combobox
TLB_CUT	Clipboard cut
TLB_COPY	Clipboard copy
TLB_PASTE	Clipboard paste
TLB_SPACER	Space between toolbar icons
TLB_NEW	File New
TLB_OPEN	File Open
TLB_SAVE	File Save
TLB_PRINT	Print
TLB_HELP	Help
TLB_PAR	Show paragraph markers
TLB_PREVIEW	Print preview
TLB_NUMBER	Set paragraph numbering
TLB_BULLET	Set paragraph bullet
TLB_UNDO	Undo
TLB_REDO	Redo
TLB_FIND	Text search



Image image // The image object to draw the toolbar icon. The toolbar icon consists of 24x24 pixels. Therefore, the picture must be at least 24 pixels wide and 24 pixels tall. Only first 24x24 pixels are used. The remaining pixels are ignored. This parameter is used by the TerAddToolBarIcon3 method only.

String balloon; // The tool tip text string.

### Example:

```
// create a custom icon and connect it to
the ID_PARA_KEEP COMMAND

tej.TerAddToolBarIcon(0,0,tej.ID_PARA_KEEP,"pict.bmp","Keep");
tej.TerAddToolBarIcon(0,tej.TLB_SPACER,0,null,null);
tej.TerAddToolBarIcon(0,tej.TLB_LINE,0,null,null);

// add an existing 'paste' icon
tej.TerAddToolBarIcon(0,tej.TLB_SPACER,0,null,null);
tej.TerAddToolBarIcon(0,tej.TLB_PASTE,0,null,null);
tej.TerAddToolBarIcon(0,tej.TLB_SPACER,0,null,null);
tej.TerAddToolBarIcon(0,tej.TLB_LINE,0,null,null);

// refresh the toolbar
tej.TerRecreateToolBar(true);
```

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



## TerHideToolBarIcon

**Hide or redisplay a toolbar icon.**

boolean TerHideToolBarIcon(id, hide)

int id; // The id of the icon to hide. Please refer to TerEditTooltip function for a list of toolbar icon ids.



boolean hide; // Set to true to hide the icon, or set to false to redisplay a previously hidden icon.

**Description:** Please note that the changes made by this function are not displayed on an 'existing' TE control window until the toolbar is recreated using the TerRecreateToolbar function. You would typically call this function multiple times to hide more than one icons and then call the TerRecreateToolbar function once to redisplay the modified toolbar. When a new control is created, it would automatically hide the icons flagged by this function.

**Return Value:** This function returns a true value if successful.

**See Also:**

[TerRecreateToolbar](#)

[TerEditTooltip](#)



## TerRecreateToolbar

**Recreate the toolbar.**

boolean TerRecreateToolbar(show)

boolean show; // Show the toolbar after it is recreated.

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

**See Also:**

[TerHideToolbarIcon](#)



## TerSetToolbarComboWidth

**Set the width of a combo-box on the toolbar.**

boolean TerUpdateToolbar(id, width, recreate)

int id; // Set this parameter to one of the toolbar ids corresponding to a combo-box toolbar item. Please refer to TerEditTooltip function for a list of toolbar icon ids.

int width; // New width of the combo-box in screen pixels.

boolean recreate; // Set to TRUE to recreate the toolbar with new combo-width.

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



## TerUpdateToolBar

**Update toolbar.**

boolean TerUpdateToolBar()

**Description:** This function should be called to update the toolbar after calling an API function which might need refreshing of the toolbar.

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



## Undo

### In This Chapter

[TerFlushUndo](#)

[TerSetMaxUndo](#)

[TerSetUndoRef](#)



## TerFlushUndo

**Flush undo/redo buffer.**

boolean TerFlushUndo()

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

### See Also:

[TerSetUndoRef](#)



## TerSetMaxUndo

**Set maximum undo or redo levels.**

int TerSetMaxUndo( MaxUndo)

int MaxUndo; // The maximum number of undo or redo allowed. The default value is 40.

Return Value: This functions returns true when successful.

See Also: TerFlushUndo, TerSetUndoRef



## TerSetUndoRef

Set or retrieve undo reference id.

```
int TerSetUndoRef( UndoRef)
```

```
int UndoRef; // The undo reference id. Set this parameter to -1 to  
             simply retrieve the current undo reference id.
```

**Comment:** This function can be used to connect multiple API calls to one undo buffer. You would first call this function before any API is called to retrieve the current undo reference count. You would then call this function before each API call to reset the undo reference to the initial value.

**Return Value:** This function returns the previous value of the undo reference id when successful. Otherwise it returns -1.

### See Also:

[TerFlushUndo](#)  
[TerSetMaxUndo](#)



## Input Field

### In This Chapter

[TerGetCheckboxInfo](#)  
[TerGetComboboxInfo](#)  
[TerGetInputFieldInfo](#)  
[TerGetTextFieldInfo](#)  
[TerInsertCheckBoxField](#)  
[TerInsertComboBoxField](#)  
[TerInsertTextInputField](#)  
[TerLocateInputField](#)  
[TerSetCheckboxInfo](#)  
[TerSetComboboxInfo](#)  
[TerSetInputFieldInfo](#)  
[TerSetTextFieldInfo](#)



## TerGetCheckboxInfo

Retrieve the information for a checkbox input field.

```
boolean TerGetCheckboxInfo( id)
```

```
int id; // Input field id to retrieve information.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutBool method:**

```
boolean checked; // This value is true value if the checkbox is checked.
                // Otherwise it is false.
```

Upon a successful return from this method, this value can be retrieve as following:

```
checked=tej.TerGetOutBool( "IsChecked" )
```

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

#### See Also:

[TerLocateInputField](#)  
[TerGetInputFieldInfo](#)  
[TerGetTextFieldInfo](#)  
[TerSetCheckboxInfo](#)



## TerGetComboboxInfo

**Retrieve the information for a combo-box input field.**

```
int TerGetComboboxInfo( id)
```

```
int id; // Id of the combo-box field to retrieve information.
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutStr method:**

```
String items; // String containing the combo-box items. Each item in
              // the list is delimited by a '|' character.
```

Upon a successful return from this method, this value can be retrieve as following:

```
items=tej.TerGetOutStr( "Items" )
```

**Return Value:** This function returns a zero based index of the selected item when successful. A value of -1 indicates an error condition.



## TerGetInputFieldInfo

### Retrieve the common input field information.

boolean TerGetInputFieldInfo( id)

int id; // Input field id to retrieve information.

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutStr and related methods:**

String name; // The field name.

Upon a successful return from this method, this value can be retrieve as following:

```
name=tej.TerGetOutStr( "Name" )
```

int type; // The field type:

FIELD\_TEXTBOX Text box field

FIELD\_CHECKBOX Checkbox field

Upon a successful return from this method, this value can be retrieve as following:

```
type=tej.TerGetOutInt( "Type" )
```

**boolean border; // The border information. The argument returns true if the field has border, otherwise it returns a false value.**

Upon a successful return from this method, this value can be retrieve as following:

```
border=tej.TerGetOutBool( "HasBorder" )
```

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

#### See Also:

[TerLocateInputField](#)

[TerGetCheckboxInfo](#)

[TerGetTextFieldInfo](#)

[TerSetInputFieldInfo](#)



## TerGetTextFieldInfo

Retrieve the information for a textbox input field.

boolean TerGetTextFieldInfo( id)

int id; // Input field id to retrieve information.

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutStr and related methods:**

String data; // The current text data in the text box.

Upon a successful return from this method, this value can be retrieve as following:

```
data=tej.TerGetOutStr("Data")
```

int MaxChars; // The current maximum text length for the text box.

Upon a successful return from this method, this value can be retrieve as following:

```
MaxChars=tej.TerGetOutInt("MaxChars")
```

int width; // The text box width in twips.

Upon a successful return from this method, this value can be retrieve as following:

```
width=tej.TerGetOutInt("Width")
```

String typeface; // The font typeface for the text in the text box.

Upon a successful return from this method, this value can be retrieve as following:

```
typeface=tej.TerGetOutStr("Typeface")
```

int TwipsSize; // The font typeface size in twips (20 twips = 1 point).

Upon a successful return from this method, this value can be retrieve as following:

```
TwipsSize=tej.TerGetOutInt("TwipsSize")
```

int style; // The style information for the font:

**BOLD**                      Bold style

**ITALIC**                    Italic style

**ULINE**                     Underline style

Upon a successful return from this method, this value can be retrieve as following:

```
style=tej.TerGetOutInt("Style")
```

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

**See Also:**

[TerLocateInputField](#)

[TerGetInputFieldInfo](#)  
[TerGetCheckboxInfo](#)  
[TerSetTextFieldInfo](#)



## TerInsertCheckBoxField

**Insert a checkbox field at the cursor position.**

```
int TerInsertCheckBoxField( name, twipsize, checked, repaint)
```

String name;	// field name
int twipsize;	// The width of the checkbox specified in Twips unit.
boolean checked;	// Set to true to create the checkbox initially checked.
boolean Insert;	// Set to true to insert the checkbox. Set to false to create a check-box object, but do not insert it in the text.
boolean repaint;	//Repaint the window after this operation

**Description:** This function inserts a checkbox.

**Return Value:** This function returns the picture id of the field. It returns 0 to indicate an error condition.

### See Also:

[TerInsertTextField](#)  
[TerLocateInputField](#)  
[TerGetCheckboxInfo](#)



## TerInsertComboBoxField

**Insert a combo-box field at the cursor position.**

```
int TerInsertComboBoxField(name, items, SelectedItem, insert, repaint)
```

String name;	// field name
String items;	// List of items in the combo-box. Each item must be delimited by a ' ' character. Example: "Red Blue Green Yellow".
String SelectedItem;	// The value of the selected item. Example: "Blue".

```
boolean Insert;           // Set to TRUE to insert the combo-box. Set to FALSE to
                          // create a combo-box object, but do not insert it in the text.

boolean repaint;         //Repaint the window after this operation
```

**Description:** This function inserts a combo-box.

**Return Value:** This function returns the picture id of the field. It returns 0 to indicate an error condition.



## TerInsertTextField

**Insert a text input field at the cursor position.**

```
int TerInsertTextField( name, InitText, MaxLen, border, typeface, twipsize, style,
color, insert, repaint)
```

```
String name;             // field name

String InitText;        // Initial text

int MaxLen;             // Maximum number of characters allowed in the field.
                        // Set to 0 to specify no limit.

boolean border;         // Set to true to draw the box around the text field.

String typeface;       // font typeface for the field text.

int twipsize;          // Font pointsize specified in Twips unit. (20 Twips = 1
                        // Point).

int style;              // Style (BOLD, ULINE, ITALIC) for the text. Use the
                        // logical OR operator to specify more than one style
                        // constants.

Color color;           // Foreground color for the text. This argument is
                        // ineffective currently. Set to 0.

boolean insert;        // Set to true to insert the new input field into the
                        // document. Set to false to simply return the picture id of
                        // the new field without inserting into the text.

boolean repaint;       //Repaint the window after this operation
```

**Description:** This function inserts a text box. The user can input data into the text box.



**Return Value:** This function returns the picture id of the text input field. It returns 0 to indicate an error condition.

**See Also:**

[TerInsertCheckBoxField](#)  
[TerLocateInputField](#)  
[TerGetInputFieldInfo](#)  
[TerGetTextFieldInfo](#)



## TerLocateInputField

**Locate an input field in the document.**

int TerLocateInputField( location, repaint)

```
int location;                // Use one of the following constants:

                             TER_FIRST:    Search from the top of the file and
                             locate the first occurrence of the input
                             field.

                             TER_LAST:     Search from the bottom of the file and
                             locate the last occurrence of the input
                             field.

                             TER_NEXT:     Find the next occurrence of the input
                             field.

                             TER_PREV:     Find the previous occurrence of the
                             input field.

                             TER_CUR:      Current field with focus

boolean repaint;            // Set to true to repaint the screen after this operation.
```

**Return Value:** This function returns the object id of the located field when successful. Otherwise, it return 0.

**See Also:**

[TerInsertCheckBoxField](#)  
[TerGetInputFieldInfo](#)  
[TerSetInputFieldInfo](#)



## TerSetCheckboxInfo

### Set the information for a checkbox input field.

boolean TerGetCheckboxInfo( id, checked)

int id; // Input field id to retrieve information.

boolean checked; // Set to true to check the checkbox.

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

#### See Also:

[TerLocateInputField](#)

[TerSetInputFieldInfo](#)

[TerSetTextfieldInfo](#)

[TerGetCheckboxInfo](#)



## TerSetComboboxInfo

### Set the information for a combo-box input field.

boolean TerSetComboboxInfo(hWnd, id, Items, SelItemIdx, repaint)

HWND hWnd; // The handle of the window to be accessed.

int id; // Input field id to retrieve information.

String items; // A String containing a list of new combo-box items.  
Each item in the list should be delimited by a '|' character. Example: "Red|Blue|Green|Yellow"

int SelItemIdx; // A zero based index of the item in the 'items' parameter to be selected automatically. For example, to select the item 'Blue', you would pass a value of 1 for this parameter.

boolean repaint; // Set to TRUE to repaint the control after this operation.

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



## TerSetInputFieldInfo

### Set the common input field information.

```

boolean TerSetInputFieldInfo( id, name, border)

int id;                // Input field id to retrieve information.

String name;          // The new field name.

boolean border;       // Set to true to set the border around the field. This
                       // argument is applicable to the textbox input field only.

```

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

#### See Also:

[TerLocateInputField](#)  
[TerSetCheckboxInfo](#)  
[TerSetTextfieldInfo](#)  
[TerGetInputFieldInfo](#)



## TerSetTextfieldInfo

### Set new information for a textbox input field.

```

boolean TerSetTextfieldInfo(id, data, MaxChars, width, typeface, TwipsSize, style)

int id;                // Input field id to retrieve information.

String data;          // The new text data for the text box.

int MaxChars;         // The maximum text length for the text box. Set to 0 for
                       // no maximum limit.

int width;            // The text box width in twips.

String typeface;      // The font typeface for the text in the text box.

int TwipsSize;        // The font typeface size in twips (20 twips = 1 point).

int style;            // The style information for the font. Set to 0 for default.
                       // The following styles can be selected for the text box.
                       BOLD    Bold style
                       ITALIC  Italic style
                       ULINE   Underline style

                       Please use the logical OR operator to specify more than
                       one flag.

```

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

**See Also:**

[TerLocateInputField](#)  
[TerSetInputFieldInfo](#)  
[TerSetCheckboxInfo](#)



## Page

This chapter includes page manipulation functions. Please also refer to the [Section Formatting](#) page for other related functions.

**In This Chapter**

[TerGetDispPageNo](#)  
[TerGetPageBorderDim](#)  
[TerGetPageCount](#)  
[TerGetPageFirstLine](#)  
[TerGetPageNumFmt](#)  
[TerGetPageOffset](#)  
[TerGetPageParam](#)  
[TerGetPagePos](#)  
[TerGetPageSect](#)  
[TerPageBreak](#)  
[TerInsertPageRef](#)  
[TerPageFromLine](#)  
[TerPageBitmap](#)  
[TerPosPage](#)  
[TerRepaginate](#)  
[TerSetPageBkColor](#)  
[TerSetPageNumFmt](#)  
[TerSetPagePos](#)



## TerGetDispPageNo

**Get the display page number corresponding to an actual page number.**

```
int TerGetDispPageNo( PageNo)
```

```
int PageNo; // Actual page number between 0 to TotalPages-1.
```

**Return Value:** The display page number can be different from the actual page number since the display page number can be reset at the section breaks.

**See Also:**

[TerGetPagePos](#)  
[TerGetField](#)



## TerGetPageBorderDim

**Return the page border dimension.**

```
int TerGetPageBorderDim()
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt method:**

```
int width; // The page border width (left and right) in twips.  
Upon a successful return from this method, this value  
can be retrieve as following:
```

```
width=tej.TerGetOutInt("Width")
```

```
int height; // The top page border height in twips.
```

Upon a successful return from this method, this value  
can be retrieve as following:

```
height=tej.TerGetOutInt("Height")
```

**Description:** This function is available in the PageMode or PrintView mode only.

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



## TerGetPageCount

**Get the page count and current page number.**

```
boolean TerGetPageCount( out TotalPages, out CurPage)
```

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt method:**

```
int TotalPages; // The total number of pages in the document.  
Upon a successful return from this method, this value can  
be retrieve as following:
```

```
TotalPages=tej.TerGetOutInt("TotalPages  
")
```

```
int CurPage; // The current page number (Zero based)
```

Upon a successful return from this method, this value can  
be retrieve as following:

```
CurPage=tej.TerGetOutInt ("CurPage")
```

**Comment:** This function is valid in the PageMode only.

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

**See Also:**

[TerSetPagePos](#)

[TerGetDispPageNo](#)



## TerGetPageFirstLine

**Return the first line number of the specified page.**

```
int TerGetPageFirstLine( PageNum)
```

```
int PageNum; // Page number between 0 and TotalPages-1
```

**Description:** This function is available in the PageMode or PrintView mode only.

**Return Value:** This function returns the first line of the specified page number.

**See Also:**

[TerPageFromLine](#)



## TerGetPageNumFmt

**Retrieve the print format for the page number string.**

```
int TerGetPageNumFmt(sect, format)
```

```
int sect; // Section id to retrieve the page number format. You  
can also set this parameter to SECT_CUR to specify the  
current section.
```

**Return Value:** This function returns one of the following constants.

NBR\_DEC            Decimal number

NBR\_UPR\_ROMAN    Uppercase roman letters

NBR\_LWR\_ROMAN    Lowercase roman letters

NBR\_UPR\_ALPHA    Uppercase alphabets

NBR\_LWR\_ALPHA    Lowercase alphabets

A return value of -1 indicates an error condition.

**See Also**

[TerSetPageNumFmt](#)



## TerGetPageOffset

**Retrieve the visible page offset and page dimension.**

boolean TerGetPageOffset(page, rel)

int page;	The page number to retrieve the offset.
int rel;	This variable can be set to one of the following constants:
	REL_TEXT_BOX    Get offset relative to the top of the text area.
	REL_WINDOW    Get offset relative to the top of the window's client area.

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt methods:**

int x;	Page offset in the x direction in screen units. Upon a successful return from this method, this value can be retrieve as following: <code>x=tej.TerGetOutInt ("X" )</code>
int y;	Page offset in the y direction in screen units. Upon a successful return from this method, this value can be retrieve as following: <code>y=tej.TerGetOutInt ("Y" )</code>
int width;	Visible page width in screen units. Upon a successful return from this method, this value can be retrieve as following: <code>width=tej.TerGetOutInt ("Width" )</code>
int height;	Visible page height in screen units. Upon a successful return from this method, this value can be retrieve as following:

```
height=tej.TerGetOutInt ( "Height " )
```

**Return Value:** This function returns true if the page is visible.



## TerGetPageParam

**Get the page parameters.**

```
int TerGetPageParam(PageNo, type)
```

```
int PageNo;           // Page number (zero based)
```

```
int type;             // The parameter to retrieve:
```

PP_PAGE_HDR_HT	Height of the page header area in twips unit.
PP_PAGE_BODY_HT	Height of the page body area (area between the header and footer) in twips unit.
PP_PAGE_FTR_HT	Height of the page footer area in twips unit.
PP_PAGE_BODY_AVAIL_HT	Height of the available area on the body of the page in twips unit.
PP_TOP_SECT	Section id at the top of the page.
PP_FIRST_LINE	First line of the page
PP_LAST_LINE	Last line of the page
PP_DISP_NO	Display page number.

**Return Value:** The function returns the value for the requested parameter. It returns FP\_ERROR to indicate an error condition.



## TerGetPagePos

**Get the current page number and the display offset.**

```
boolean TerGetPagePos( out page, out y)
```



**Upon a successful return from this method, the following information can be retrieved by using TerGetOutInt methods:**

```
int page; // The current page number  
Upon a successful return from this method, this value can  
be retrieve as following:  
page=tej.TerGetOutInt("CurPage")  
  
int y; // The offset of the top of the window relative to the top of  
the page. This value is returned in twips unit.  
Upon a successful return from this method, this value can  
be retrieve as following:  
y=tej.TerGetOutInt("PageOffset")
```

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

**See Also:**

[TerGetPageCount](#)  
[TerSetPagePos](#)  
[TerGetDispPageNo](#)



## TerGetPageSect

**Return the section id for a page.**

```
int TerGetPageSect( PageNum)
```

```
int PageNum; // Page number between 0 and TotalPages-1
```

**Description:** This function is available in the PageMode or PrintView mode only.

**Return Value:** This function returns the section number containing the first line of a page.



## TerPageBreak

**Create a hard page break.**

```
boolean TerPageBreak(repaint)
```

```
boolean repaint; //Repaint the window after this operation
```

**Description:** This function is used to place the following text on the new page. The page break is created before the current line. Please note that a page break can not be created inside an object such as table, frame, text box, etc.

A page break is indicated by a solid line.

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

**See Also:**

[TerColBreak](#)

[TerSectBreak](#)



## TerInsertPageRef

**Insert page reference field.**

boolean TerInsertPageRef(bookmark, IsHyperlink, IsAlphabetic, repaint)

String bookmark; // The name of the bookmark to build a reference. The bookmark must already exist in the document. You can use the TerInsertBookmark function to insert a bookmark.

boolean IsHyperlink; // Set to True to treat this page reference as a hyperlink. When the user clicks on this page-reference hyperlink, the cursor will jump to the referenced bookmark.  
The hyperlink cursor must be turned on using the ID\_SHOW\_HYPERLINK\_CURSOR command to display the hyperlink cursor.

boolean IsAlphabetic; // Set to True to display the page number in the alphabetic format.

boolean repaint; //Repaint the window after this operation

**Comment:** The page-reference field displays the page number where the referenced bookmark is located.

**Return Value:** This function returns a True value when successful, otherwise it returns a false value.

**See Also**

[TerInsertBookmark](#)

[TerInsertToc](#)



## TerPageFromLine

### Retrieve the page number containing the given line.

`int TerPageFromLine(LineNo)`

`int LineNo; // Line number (0 to TotalLines - 1)`

**Return Value:** This function returns the page number that contains the given text line number. A value of -1 indicates an error condition.

#### See Also:

[TerGetPageFirstLine](#)



## TerPageBitmap

### Retrieve the image of a page of text.

`Bitmap TerPageBitmap(PageNo)`

`int PageNo; // Page number number (0 to TotalPages - 1)`

**Return Value:** This function returns the Bitmap object containing the text for the page. A null value indicates an error condition.

When creating bitmaps for multiple pages, it is efficient to sandwich multiple calls to the `TerPageBitmap` function between the calls to the `TerSetPrintPreview` function:

```
int TotalPages=tej.TerGetParam(tej.TP_TOTAL_PAGES);

tej.TerSetPrintPreview(true); // start the document bitmap
                               output

for (int page=0;page<TotalPages;page++) {

    tej.TerPageBitmap(page);

    TotalPages=tej.TerGetParam(tej.TP_TOTAL_PAGES);
    // this line is needed only if the Tej control is NOT
    set to use page-mode
}

tej.TerSetPrintPreview(false);
    // end the document bitmap output
```



## TerPosPage

**Position at the specified page number.**

boolean TerPosPage ( NewPage)

intnew Page; // Page number (0 to TotalPages - 1) to position at.

**Description:** This function is available in the Page Mode only.

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

### See Also:

[TerPosTable](#)

[TerGetSeqSect](#)



## TerRepaginate

**Repaginate the document.**

boolean TerRepaginate( repaint)

boolean repaint; // true to repaint the document after this operation

**Description:** This function rewraps and repaginates a document. This function should be used instead of the TerRewrap function in the PrintView, Page mode, and Fitted View modes.

**Return Value:** The function returns true when successful

### See Also:

[TerRewrap](#)

[TerReformatTable](#)



## TerSetPageBkColor

**Set page background color.**

boolean TerSetPageBkColor(BkColor)

Color BkColor; //The background color for the page.

or, The color value can be passed as a Color object, or an HTML color string such as red, "#ff0000".  
String HtmlColor;

**Description:** This function is used set a background color for the page. If the page borders are active, then the color is applied to the area inside the page borders.

Use the TerGetParam function to retrieve the current page background color.

**Example:**

```
tejl.TerSetPaegBkColor(Color.Red); // set page color to red
tejl.TerRepaint(false); // repaint to show the color
```

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



## TerSetPageNumFmt

**Set the print format for the page number string.**

boolean TerSetPageNumFmt(sect, format)

int sect; // Section id to apply changes. You can also set this parameter to SECT\_CUR to edit the current section, or set it to SECT\_ALL to apply changes to all sections in the document.

int format; // The 'format' parameter can be set to one of the following constants:

NBR\_DEC                  Decimal number

NBR\_UPR\_ROMAN          Uppercase roman letters

NBR\_LWR\_ROMAN          Lowercase roman letters

NBR\_UPR\_ALPHA          Uppercase alphabets

NBR\_LWR\_ALPHA          Lowercase alphabets

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

**See Also**

[TerGetPageNumFmt](#)



## TerSetPagePos

**Position at the specified page number and at the specified display offset with the page.**

boolean TerSetPagePos( page, y)

int page; // This variable specifies the page number (0 to TotalPages -1) to position at.

int y; // This variable specifies the offset of the top of the window relative to the top of the page. This value is specified in the twips unit.

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

**See Also:**

[TerGetPagePos](#)



## Search, Replace, Locate

**In This Chapter**

[TerLocateFieldChar](#)

[TerSearchReplace](#)

[TerSetSearchString](#)



### TerLocateFieldChar

**Locate the character with the given field id.**

boolean TerLocateFieldChar(FieldId,FieldCode, present, StartLine, StartCol, forward)

int FieldId; // Select one of the following field ids to locate:

FIELD\_NAME: Mail-merge field name

FIELD\_DATA: Mail-merge field data

FIELD\_HLINK: Hyperlink field

FIELD\_TEXTBOX Input t text field

String FieldCode; // Field code associated with Field Id. Pass a null value for the field ids (such as FIELD\_NAME and FIELD\_DATA) that do not use field code.

boolean present;	// true to test for the presence of the given field id, or false to test for the absence of the given field id.
int StartLine;	(INPUT/OUTPUT) Specifies the variable to the line number to start the search.  Upon a successful search, the line number of the located text be retrieved as following:  <code>line=tej.TerGetOutInt("StartLine")</code>
int StartCol;	(INPUT/OUTPUT) Specifies the variable to the column number to start the search.  Upon a successful search, the column number of the located text be retrieved as following:  <code>col=tej.TerGetOutInt("StartCol")</code>
boolean forward;	// true to scan the text in the forward direction, or false to scan the text in the backward direction.

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

**See Also:**  
[TerLocateField](#)



## TerSearchReplace

### Search, replace or retrieve text.

boolean TerSearchReplace(search, replace, flags, StartPos, pEndPos)

Int TerSearchReplace2( search, replace, flags, StartPos, EndPos)

String search;	// search text string
String replace;	// replace text string
int flags;	// mode flags
int StartPos;	// start text position
int pEndPos;	// variable to the end text position. This parameter is not used by the TerSearchReplace2 function.
int EndPos;	// The end position of the String to replace. This parameter is not used by the TerSearchReplace function.

```
int BufSize; // variable to the size of the retrieved text. This parameter
              is not used by the TerSearchReplace2 function.
```

Description: This function has three operational modes: search, replace and retrieve.

**Search Mode:** To initiate this mode, set the SRCH\_SEARCH flag in the 'flags' parameter. You can also set the following bits in the 'flags' parameter:

```
SRCH_CASE           Case sensitive search
SRCH_WORD           Match whole words
SRCH_SCROLL         Scroll the located text into view.
SRCH_SKIP_HIDDEN_TEXT Skip over the hidden text
SRCH_BACK           Search in the backward direction
SRCH_NO_REPLACE_PROT_TEXT Do not replace protected text.
```

The search String is passed via the 'search' parameter. The search String consists of regular text and certain special characters. The special characters are inserted using the '^' prefix:

```
^p Paragraph character
^t Tab character
^m Manual page break
^b Section break
^+ Em dash
^- En dash
^^ ^ character.
```

The initial search location is specified by the 'StartPos' parameter. This parameter specifies the absolute character position since the beginning of the file.

If the search String is located using the *TerSearchReplace* function, the absolute character position of the located text can be retrieved as following:

```
EndPos=tej.TerGetOutInt("EndPos")
```

If the search String is located using **TerSearchReplace2** function, then the absolute character position is returned via function return value. If the String is not found, then this function returns -1.

**Replace Mode:** To initiate this mode, set the SRCH\_REPLACE flag in the 'flags' parameter. This function replaces the text between the 'StartPos' and 'pEndPos' (or EndPos for the TerSearchReplace2 function) absolute character positions by the specified replacement text. The replacement text is specified by the 'replace' argument.

**Retrieve Mode:** To initiate this mode, set the SRCH\_RETRIEVE flag in the 'flags' parameter. This function retrieves the text between the 'StartPos' and 'EndPos' absolute character positions. This function returns even the hidden text. The text between the StartPos and EndPos can be subsequently retrieved upon the successful return from this method as following

```
String RetrievedText=tej.TerGetOutStr("Search")
```



### Return Value for the TerSearchReplace function:

This function returns true if successful. Otherwise, it returns a false value.

### Return Value for the TerSearchReplace2 function:

When the SRCH\_SEARCH flag is specified, the return value indicates the absolute character position of the String found. It returns -1 if the String is not found.

When the SRCH\_REPLACE flag is specified, this function returns true if successful. Otherwise, it returns a false value

### Example:

```
pos=0;

while (pos>=0) {
    pos=tej.TerSearchReplace2(SearchString, "",
        tej.SRCH_SEARCH+tej.SRCH_WORD+ tej.SRCH_CASE, pos,
0);

    if (pos>=0) tej.TerSearchReplace2("", ReplaceString,
        tej.SRCH_REPLACE, pos, pos+SearchString.Length-1);
}
```



## TerSetSearchString

**Set the String for backward and forward search menu commands.**

```
boolean TerSetSearchString( SearchFor, CaseSensitive)
```

```
String SearchFor;           // search string
```

```
boolean CaseSensitive;     // Set to true for the case sensitive search.
```

**Return Value:** The function returns true when successful.



## Track Changes

This chapter includes document modification tracking functions. TE can track modification made by multiple reviewers. You can search for each modified text string. You can accept individual modification, or you can accept all modification at once. The process of acceptance merges the modification to the main document.

#### In This Chapter

[TerAcceptChanges](#)  
[TerDeleteReviewer](#)  
[TerEnableTracking](#)  
[TerFindNextChange](#)  
[TerGetReviewerInfo](#)  
[TerRejectChanges](#)



## TerAcceptChanges

### Accept modified text.

```
int TerAcceptChanges(all, msg, repaint)
```

```
boolean all;           // Set to TRUE to accept all changes. Set to false to  
                        // accept the current change.
```

```
boolean msg;          // Set to true to display confirmation and completion  
                        // messages.
```

```
boolean repaint;      // Repaint the window after this operation
```

**Description:** This function merges the modified text to the document. The 'deleted' text is removed from the document. The 'inserted' text attribute is set to normal.

The tracking mode should be turned off to enable the use of this function.

The ID\_ACCEPT\_CHANGE and ID\_ACCEPT\_ALL\_CHANGES ids can also be used with the TerCommand function or the Command property to accept modified text.

**Return Value:** This function returns the number of text strings accepted. It returns -1 to indicate an error condition.



## TerDeleteReviewer

### Delete a reviewer

```
boolean TerDeleteReviewer(RevId)
```

```
int RevId; // Reviewer id to delete
```

**Comment:** This function deletes the specified reviewer id from the current document. The reviewer id should be deleted only if the document does not contain any changes associated with the reviewer.

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



## TerEnableTracking

### Enable tracking of document modification.

```
boolean TerEnableTracking(enable, name, UseDefaultClrStyle, InsStyle, InsColor,  
DelStyle, DelColor)
```

```
boolean enable; // Set to TRUE to enable tracking. Set to false to disable  
tracking.
```

```
String name; // The reviewer name. Set to "" to assume currently  
logged user name.
```

```
boolean UseDefaultClrStyle; // Use the default value for color and style for the deleted  
and inserted text. When this parameter is set to TRUE,  
the following color and style parameters are ignored.
```

```
int InsStyle; // The style to apply to the newly inserted text. Please  
refer to the SetTerCharStyle function for the list of  
available character style.
```

This parameter is ignored if the UseDefaultClrStyle parameter is set to TRUE.

```
Color InsColor; // The color to apply to the newly inserted text.  
or,  
String InsColor; The color value can be passed as a Color object, or an  
HTML color string such as red, "#ff0000".
```

This parameter is ignored if the UseDefaultClrStyle parameter is set to TRUE.

```
int DelStyle; // The style to apply to the deleted text. Please refer to  
the SetTerCharStyle function for the list of available  
character style.
```

This parameter is ignored if the UseDefaultClrStyle parameter is set to TRUE.

```
Color DelColor; // The color to apply to the deleted text.  
or,  
The color value can be passed as a Color object, or an
```

String DelColor;                      HTML color string such as red, "#ff0000".  
This parameter is ignored if the UseDefaultClrStyle parameter is set to TRUE.

**Comment:** The ID\_TRACK\_CHANGES id can also be used with the TerCommand method or the Command property to enable/disable tracking of modification.

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



## TerFindNextChange

**Find next or previous modified text.**

boolean TerFindNextChange(forward, repaint)

boolean forward;                      // Set to TRUE to find the next modified string. Set to false to find the previous modified string.

boolean repaint;                      //Repaint the window after this operation

**Comment:** The ID\_NEXT\_CHANGE and ID\_PREV\_CHANGE ids can also be used with the TerCommand function or the Command property to locate modified text.

**Return Value:** This function returns TRUE if the String is located, otherwise it returns FALSE.



## TerGetReviewerInfo

**Retrieve reviewer info**

boolean TerGetReviewerInfo(hWnd, RevId)

int RevId;                              // Reviewer id to retrieve information

**Upon a successful return from this method, the following information can be retrieved by using TerGetOutStr methods:**

String RevName;                      // The reviewer name

Upon a successful return from this method, this value can be retrieve as following:

```
RevName=tej.TerGetOutStr("Name")
```

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



## TerRejectChanges

**Reject modified text.**

```
int TerRejectChanges(all, msg, repaint)
```

boolean all; // Set to TRUE to reject all changes. Set to false to reject the current change.

boolean msg; // Set to true to display confirmation and completion messages.

boolean repaint; // Repaint the window after this operation

**Description:** This function annuls the modification to the document. The 'inserted' text is removed from the document. The 'deleted' text is reinstated in the document.

The tracking mode should be turned off to enable the use of this function.

The ID\_REJECT\_CHANGE and ID\_REJECT\_ALL\_CHANGES ids can also be used with the TerCommand function or the Command property to reject modified text.

**Return Value:** This function returns the number of text strings rejected. It returns -1 to indicate an error condition.



## Comment



### TerApplyComment

**Apply comment to the selected text.**

```
int TerApplyComment(ShowDialog, AuthorName, AuthorInitials, CommentText, repaint)
```

boolean ShowDialog; // Set to TRUE to show a dialog to accept the author information and the comment text. Set to false to use the information from the remaining parameters passed to

```

this method

String AuthorName;      // The author name for the comment

String AuthorInitials;  // Author initials

String CommentText;     // Comment text. Use the cr/lf sequence to include a
                        // paragraph break.

boolean repaint;        // refresh the screen after this operation

```

**Comment:** Some text must be selected before calling this method. The comment is applied to the selected text. The selected text is highlighted and pointed to the comment text in the right-margin area. The comments are displayed horizontally when the right-margin is sufficiently wide. Otherwise, it is displayed vertically.

**Return Value:** This function returns the comment-id when successful, otherwise it returns a -1 value.



## TerEditComment

### Edit or delete an existing comment.

```
boolean TerEditComment(ShowDialog, id, pAuthorName, pAuthorInitials,
pCommentText, repaint)
```

```

boolean ShowDialog;      // Set to TRUE to show a dialog to accept the new author
                        // information and the comment text. Set to false to use the
                        // information from the remaining parameters passed to
                        // this method

int id;                  // Comment id to modify

String AuthorName;      // The new author name for the comment

String AuthorInitials;  // The new author initials

String CommentText;     // New Comment text. Use the cr/lf sequence to include a
                        // paragraph break.
                        // Set to "" to delete the comment.

boolean repaint;        // refresh the screen after this operation

```

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



## Menu Command

### In This Chapter

[TerCommand](#)

[TerMenuEnable](#)

[TerMenuSelect](#)

[TerIgnoreCommand](#)



## TerCommand

### Execute a TEJ menu command.

```
boolean TerCommand( CommandId)
```

```
boolean TerCommand2( CommandId, send)
```

```
int CommandId; // Please refer to the 'command' property under the  
Control Properties chapter for a list of available  
command ids.
```

```
boolean send; // Set to true to execute the command immediately, or  
set to false to 'post' the command for delayed execution.
```

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



## TerMenuEnable

### Get menu item 'enable' status

```
int TerMenuEnable( MenuId)
```

```
boolean TerMenuEnable2( MenuId)
```

```
int MenuId; // menu id. Please refer to the 'command' property for a  
list of menu (or command) ids.
```

**Description:** If your program creates a menu outside the editor window, you can use this function to test if a menu item should be enabled or grayed.

Return Value: The TerMenuEnable function returns one of the following constants:

MF\_ENABLED = Enable menu item

MF\_GRAYED = Gray out the menu item

The TerMenuEnable2 function returns true when a menu item is enabled. Otherwise it returns a false value.

Example:

```
boolean status;
```

```
status = tej.TerMenuEnable2(tej.ID_CUT);
```

```
// The 'status' variable will be true if a text block is highlighted to be copied to the clipboard.
```

**See Also:**

[TerMenuSelect](#)



## TerMenuSelect

### Get menu item selection status

```
int TerMenuSelect( MenuId)
```

```
boolean TerMenuSelect2( MenuId)
```

```
int MenuId; // menu id. Please refer to the 'command' property for a list of menu (or command) ids.
```

**Description:** If your program creates a menu outside the editor window, you can use this function to test if a menu item should be *checked*.

**Return Value:** The TerMenuSelect function returns one of the following constants:

MF\_CHECKED = Check the menu item

MF\_UNCHECKED = Uncheck the menu item

The TerMenuSelect2 function returns a true value if a menu item is to be checked. Otherwise it returns a false value.

Example:

```
boolean status;
```

```
status = tej.TerMenuSelect2(tej.ID_BOLD);
```

```
// The 'status' variable will be true if the current character has the bold style.
```

**See Also:**

[TerMenuEnable](#)



## TerIgnoreCommand

**Ignore the current preprocess command.**

```
boolean TerIgnoreCommand()
```



**Description:** This function can be used while processing the 'Preprocess' event. This function sets a flag which instructs the editor to skip processing the current command.

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



## Screen Drawing

### In This Chapter

[TerEnableRefresh](#)  
[TerGetTextHeight](#)  
[TerRepaint](#)  
[TerRewrap](#)  
[TerScrLineHeight](#)  
[TerSetBorderColor](#)  
[TerSetBorderLineColor](#)  
[TerSetFocus](#)  
[TerSetWrapWidth](#)  
[TerSetZoom](#)  
[TerSetStatusColor](#)



### TerEnableRefresh

**Enable or disable screen refresh.**

```
boolean TerEnableRefresh(enable)
```

```
boolean enable;           // True to enable the screen refresh.
```

**Description:** This function is used to enable or disable the screen painting.

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



### TerGetTextHeight

**Return the total text height in twips.**

```
int TerGetTextHeight()
```

**Description:** This function returns the total body text height of all pages in the document. The body text height does not include the header/footer text, or the table header text.

**Return Value:** This function returns the text height in twips.

#### See Also

[TerScrLineHeight](#)



## TerRepaint

**Repaint the TEJ control.**

```
boolean TerRepaint( ClearBackground)
```

```
boolean ClearBackground; // true to clear the background before repainting
```

**Description:** This function repaints every aspect of the TEJ control.

**Return Value:** The function returns true when successful



## TerRewrap

**Word wrap the entire document on demand.**

```
int TerRewrap()
```

**Description:** This function can be used to rewrap the entire document on demand.

**Return Value:** The function returns true when successful.

#### See Also:

[TerRepaginate](#)



## TerScrLineHeight

**Return the line height in screen pixels.**

```
int TerScrLineHeight( line)
```

```
int line; // line number (0 to TotalLines -1) to return the height for.
```

**Return Value:** This function returns the line height in the screen pixels.

#### See Also

[TerGetTextHeight](#)



## TerSetBorderColor

**Set the color of the border area around the text window.**

boolean TerSetBorderColor( color)

Color color; // Color of the border area. The border color gets reset  
or, when you set the text background color using the  
SetTerFields function.  
String color; The color value can be passed as a Color object, or an  
HTML color string such as red, "#ff0000".

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



## TerSetBorderLineColor

**Set the color of the border line around the page in the page-layout mode.**

boolean TerSetBorderLineColor( color)

Color color; // Color of the border area.  
or, The color value can be passed as a Color object, or an  
String color; HTML color string such as red, "#ff0000".

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



## TerSetFocus

**Set the focus cursor.**

boolean TerSetFocus()

**Description:** Normally, when the editor window is activated, the caret (cursor) shows up in the editor window. However, within certain programming environment, the caret does not appear in the editor window automatically. This function can be used to display the caret in this situation.

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



## TerSetWrapWidth

### Set the wrap width.

```
boolean TerSetWrapWidth( WidthChars, WidthTwips, repaint)
```

```
int WidthChars;           // This parameter specifies the maximum number of
                           // characters allowed in a line to trigger word wrapping.
```

```
int WidthTwips;           // This parameter specifies the length of the line in twips
                           // for word wrapping. The actual width of the text line is
                           // calculated as following:
```

```
WidthTwips - Left Margin - Left Paragraph Indentation -
Right Margin - Right Paragraph Indentation.
```

```
boolean repaint;         // true to repaint after this operation.
```

**Description:** This function is available in simple word-wrap mode only (not available when the PrintView or Page Mode is turned on). To specify the wrap width in terms of the number of characters, set the WidthTwips parameter to 0. To specify the wrap width in terms of 'twips', set the WidthChars parameter to 0. To reset to regular word wrapping, set both the WidthChars and WidthTwips parameters to 0.

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



## TerSetZoom

### Set the zoom percentage.

```
int TerSetZoom( ZoomPercent)
```

```
int ZoomPercent;        // Specify a value between 10 and 1000, the 100 being
                           // the normal display. You can set this argument to -2 to
                           // simply retrieve the current zoom percent with changing
                           // it.
```

**Return Value:** This function returns the previous zoom percent. It returns -1 if an error occurs.



## TerSetStatusColor

### Set the color of the status, ruler and toolbar area around the text window.

boolean TerSetStatusColor( color, BkColor)

Color color;	// The foreground color for the status bar text.
or,	The color value can be passed as a Color object, or an HTML color string such as red, "#ff0000".
String color;	
Color BkColor;	// The background color for the status, ruler and toolbar area.
or,	
String BkColor;	The color value can be passed as a Color object, or an HTML color string such as red, "#ff0000".

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



## Spell Checking

SpellTime must be installed to use these functions. The SpellTime.DLL and the dictionary files should be placed where the tej.dll file is located.

**Note for HTML/Javascript version:** Copy SpellTimeh5.js file to the server folder containing terh31.js. Copy the dictionary files to the same folder.

After a Tej control is created, please call the TerSetStLicenseKey function to set the product key for SpellTime.

You can use the ID\_SPELL (not available for TE for HTML/Javascript) and ID\_AUTO\_SPELL commands to use on-demand or as-you-type spell-checking.

You can also use the TerSpellCheck function to invoke on-demand spell-checking.

If you need the spell-time dictionaries to be accessed from a different directory, then create an instance of SpellTime class providing the dictionary path to the constructor. For this, you will need to include a reference to SpellTime.dll in your project. Once you get the SpellTime object, you can pass the SpellTime object to the current instance of Tej using the TerInitSpellTime function. This technique can also be used to share one instance of SpellTime with more than one instances of Tej. By default, each instance of Tej creates its own SpellTime object.

### In This Chapter

[TerInitSpellTime](#)

[TerSetStLicenseKey](#)

[TerSpellCheck](#)



## TerInitSpellTime

### Initialize SpellTime.

```
boolean TerInitSpellTime(st)
```

```
object st; // SpellTime object
```

**Description:** Normally TE automatically creates an instance of SpellTime class if SpellTime.DLL and the dictionary files are installed. However, you can use this function to override the SpellTime object created by the editor.

The following example creates an instance of SpellTime specifying a dictionary file path. It then pass this object to TE.

```
SpellTime st = new SpellTime("c:\mydirectory\dict35.d",0)
tej1.TerInitSpellTime(st)
```

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



## TerSetStLicenseKey

### Set the license key for SpellTime.

```
boolean TerSetStLicenseKey( LicenseKey)
```

```
String LicenseKey // The license key for SpellTime is e-mailed to you after
your order for SpellTime is processed.
```

The license key for SpellTime is e-mailed to you after your order for SpellTime is processed. You would set the license key for SpellTime using the TerSetStLicenseKey static function. This should be done after creating an instance of the editor control.

```
Tej.TerSetStLicenseKey("xxxxx-yyyyy-zzzzz")
```

Replace the 'xxxxx-yyyyy-zzzzz' by your license key for SpellTime. Please note that the your license key for Tej is not valid for SpellTime.

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



## TerSpellCheck

Invoke the spell checker.

```
boolean TerSpellCheck( StopAfterFirst, msg)
```

```
boolean StopAfterFirst;           // true to stop the spell check session after the first  
                                  misspelled word is found.
```

```
boolean msg;                       // true to display the termination message indicating the  
                                  number of misspelled words.
```

**Return Value:** This function returns true if document contains no misspelled words. It returns false if the StopAfterFirst parameter is set to true and a one misspelled word is found. Otherwise, the false return values indicates one or more misspelled words. A false value is also returned when SpellTime is not installed.



## HTML Add-on Interface

\*\*\*\*\*

**HTML Add-on is NOT available for TE Edit Control for Java yet. The methods described in this topic are for future use only (not available at present).**

\*\*\*\*\*

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 htn.dll to the application directory containing tej.dll.

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

```
tej1.TerSetHtnLicenseKey("xxxxx-yyyyy-zzzzz")
```

### Export TE document as HTML:

First set the output format to SAVE\_HTML using the TerSetOutputFormat method:

```
tej1.TerSetOutputFormat(tej.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=tej1.Data
```

### Import an HTML document into TE:

First set the input format as html:

```
tej1.TerSetFlags4(True,tej.TFLAG4_HTML_INPUT)
```

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

```
tej1.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 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(tej1) // pass the Tej 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 for HTML Add-on is processed.

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

// Now pass the Htn object to TE.

```
tej1.TerSetHtnObject(htn1)
```

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

#### In This Chapter

[TerSetHtnLicenseKey](#)

[TerSetHtnObject](#)



## TerSetHtnLicenseKey

### Set the license key for HTML Add-on.

```
void TerSetHtnLicenseKey( LicenseKey)
```

```
String LicenseKey // Your license key for HTML Add-on is e-mailed to you
                  after your order for HTML Add-on is processed.
```

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

### Example:

```
tej1.TerSetHtnLicenseKey("xxxxx-yyyyy-zzzzz")
```

Replace the 'xxxxx-yyyyy-zzzzz' by your license key for HTML Add-on. Please note that the your license key for Tej is *not* valid for HTML Add-on.



**Return Value:** None.

#### See Also

[TerSetHtnObject](#)



## TerSetHtnObject

**Pass an HTML Add-on object to TE.**

```
boolean TerSetHtnObject(htn)
```

```
object htn; // Htn object
```

**Description:** Normally TE automatically creates an instance of Htn class if HTN.DLL file is installed. However, you can use this function to override the Htn object created by the editor. This is useful when you need to call Htn methods (such as HtsSetFlags) prior to doing import or export from TE.

Example:

First create an Htn objec:

```
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(tej1) // pass the Tej class object to bind  
to the  
new Htn object
```

// set the license key. The license key for HTML Add-on is *e-mailed to you after your order for HTML Add-on is processed.*

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

// Now call any Htn method needed. Example:

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

// Now pass the Htn object to TE.

```
tej1.TerSetHtnObject(htn1)
```

Now TE can use this Htn object to do import or export of HTML data using the SAVE\_HTML and TFLAG4\_HTML\_INPUT flags.

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

#### See Also

[TerSetHtnLicenseKey](#)



## Miscellaneous

### In This Chapter

[TerAnd](#)  
[TerEnableSpeedKey](#)  
[TerGetBufferGr](#)  
[TerGetLastMessage](#)  
[TerGetOutBool](#)  
[TerGetOutColor](#)  
[TerGetOutInt](#)  
[TerGetOutMediaTray](#)  
[TerGetOutIntArray](#)  
[TerGetOutStr](#)  
[TerGetOutObject](#)  
[TerGetOutRect](#)  
[TerInsertUserField](#)  
[TerOr](#)  
[TerResetLastMessage](#)  
[TerSetCtlColor](#)  
[TerSetCustomMessage](#)  
[TerSetLicenseKey](#)  
[TerSetUserField](#)  
[TerUpdateDynField](#)



## TerAnd

**Return bitwise 'AND' value of two variables.**

```
int TerAnd(var1, var2)
```

```
int var1;                // First variable.
```

```
Int var2;                // Second variable
```

**Description:** This function is useful in the programming environments which does not provide a built-in bitwise AND operator.

**Return Value:** The function returns the bitwise AND value of the input variables.

### See Also:

[TerOr](#)



## TerEnableSpeedKey

**Enable or disable a speed key.**

boolean TerEnableSpeedKey( CommandId, enable)

```
int CommandId;           // Id of the command to set the key for. Here are the list of
                          // command ids and corresponding speed-key combination:
ID_PGUP                  :Keys.PageUp
ID_PGUP                  :Keys.PageUp|Keys.Shift
ID_PGDN                  :Keys.PageDown
ID_PGDN                  :Keys.PageDown|Keys.Shift
ID_UP                    :Keys.Up
ID_UP                    :Keys.Up|Keys.Shift
ID_DOWN                  :Keys.Down
ID_DOWN                  :Keys.Down|Keys.Shift
ID_LEFT                  :Keys.Left
ID_LEFT                  :Keys.Left|Keys.Shift
ID_RIGHT                 :Keys.Right
ID_RIGHT                 :Keys.Right|Keys.Shift
ID_LINE_BEGIN            :Keys.Home
ID_LINE_BEGIN            :Keys.Home|Keys.Shift
ID_LINE_END              :Keys.End
ID_LINE_END              :Keys.End|Keys.Shift
ID_CTRL_UP               :Keys.Up|Keys.Control
ID_CTRL_UP               :Keys.Up|Keys.Control
                          |Keys.Shift
ID_CTRL_DOWN             :Keys.Down|Keys.Control
ID_CTRL_DOWN             :Keys.Down|Keys.Control
                          |Keys.Shift
ID_FILE_BEGIN            :Keys.PageUp|Keys.Control
ID_FILE_BEGIN            :Keys.PageUp|Keys.Control
                          |Keys.Shift
ID_FILE_BEGIN            :Keys.Home|Keys.Control
ID_FILE_BEGIN            :Keys.Home|Keys.Control
                          |Keys.Shift
ID_FILE_END              :Keys.PageDown
                          |Keys.Control
ID_FILE_END              :Keys.PageDown
                          |Keys.Control|Keys.Shift
```

ID_FILE_END	:Keys.End Keys.Control
ID_FILE_END	:Keys.End Keys.Control  Keys.Shift
ID_NEXT_WORD	:Keys.Right Keys.Control
ID_NEXT_WORD	:Keys.Right Keys.Control  Keys.Shift
ID_PREV_WORD	:Keys.Left Keys.Control
ID_PREV_WORD	:Keys.Left Keys.Control  Keys.Shift
ID_DEL_PREV_WORD	:Keys.Back Keys.Control
ID_DEL	:Keys.Delete
ID_BACK_SPACE	:Keys.Back
ID_BACK_TAB	:Keys.Tab Keys.Shift
ID_CTRL_TAB	:Keys.Tab Keys.Control
ID_TAB	:Keys.Tab
ID_TAB	:Keys.T Keys.Control
ID_HILIGHT_LINE	:Keys.F8
ID_SELECT_ALL	:Keys.A Keys.Control
ID_CUT	:Keys.X Keys.Control
ID_CUT	:Keys.Delete Keys.Shift
ID_COPY	:Keys.C Keys.Control
ID_COPY	:Keys.Insert Keys.Control
ID_PASTE	:Keys.V Keys.Control
ID_PASTE	:Keys.Insert Keys.Shift
ID_PICT_FROM_FILE	:Keys.F8 Keys.Alt
ID_BLOCK_COPY	:Keys.C Keys.Alt
ID_BLOCK_MOVE	:Keys.M Keys.Alt
ID_SEARCH	:Keys.F5
ID_SEARCH_FOR	:Keys.F Keys.Control
ID_SEARCH_BACK	:Keys.F Keys.Control  Keys.Shift
ID_REPLACE	:Keys.F6
ID_TER_HELP	:Keys.F1
ID_SPELL	:Keys.F7 (not available for TE for HTML/Javascript)
ID_UNDO	:Keys.F8 Keys.Shift
ID_UNDO	:Keys.Back Keys.Alt
ID_UNDO	:Keys.Z Keys.Control
ID_REDO	:Keys.Y Keys.Control

```

ID_INSERT      :Keys.Insert
ID_SAVE        :Keys.F3
ID_SAVEAS     :Keys.F3|Keys.Shift
ID_QUIT        :Keys.F3|Keys.Control
ID_PRINT       :Keys.F4
ID_PRINT_OPTIONS:Keys.F4|Keys.Shift
ID_JUMP        :Keys.F10
ID_CHAR_NORMAL :Keys.D0|Keys.Alt
ID_BOLD_ON     :Keys.B|Keys.Control
ID_ULINE_ON    :Keys.U|Keys.Control
ID_ULINED_ON   :Keys.D|Keys.Control
ID_ITALIC_ON   :Keys.I|Keys.Control
ID_HIDDEN_ON   :Keys.H|Keys.Control
ID_HLINK_ON    :Keys.H|Keys.Alt
ID_PROTECT_ON  :Keys.D3|Keys.Alt
ID_SUPSCR_ON   :Keys.D4|Keys.Alt
ID_SUBSCR_ON   :Keys.D5|Keys.Alt
ID_STRIKE_ON   :Keys.D6|Keys.Alt
ID_COLOR       :Keys.D7|Keys.Alt
ID_FONTS       :Keys.F10|Keys.Alt
ID_CENTER      :Keys.D8|Keys.Alt
ID_RIGHT_JUSTIFY:Keys.D9|Keys.Alt
ID_LEFT_INDENT :Keys.L|Keys.Alt
ID_RIGHT_INDENT:Keys.R|Keys.Alt
ID_HANGING_INDENT:Keys.T|Keys.Alt
ID_BULLET     :Keys.B|Keys.Alt
ID_PARA_NBR    :Keys.N|Keys.Alt
ID_EDIT_STYLE  :Keys.S|Keys.Alt
ID_CHAR_STYLE  :Keys.D1|Keys.Alt
ID_PARA_STYLE  :Keys.D2|Keys.Alt
ID_PAGE_BREAK  :Keys.Enter|Keys.Control
ID_RETURN      :Keys.Enter
ID_RETURN      :Keys.Enter|Keys.Shift
ID_NEXT_CHANGE :Keys.N|Keys.Control
ID_PREV_CHANGE :Keys.P|Keys.Control

```

boolean enable

// True to enable or False to disable the speed key. To disable the entire accelerator table, please refer to the TerSetFlags function.

**Return Value:** The function returns the previous status of the speed key



## TerGetBufferGr

**Get the handle of the buffer Graphics object.**

```
Graphics TerGetBufferGr()
```

**Return Value:** This function returns the integer buffer Graphics object. This Graphics object is used internally to create a temporary image of the screen before actually transferring to the window. It returns null to indicate an error condition. It can also return null when the integer buffer is disabled by using the TerSetFlags function.

**Please call the GetTerFields function to retrieve the actual Graphics object of the editor window.**



## TerGetLastMessage

**Get the last message.**

```
int TerGetLastMessageCode();
```

```
String TerGetLastMessage();
```

**Return Value:** These methods return the last message generated by the editor. The TerGetLastMessageCode returns the numeric message code, and the TerGetLastMessage method returns the corresponding message text.

The return value is valid only if saving of the messages is enabled by setting the TFLAG\_RETURN\_MSG\_ID flag. This flag is set using the TerSetFlags function.

The description for the message codes can be found in the TER\_MSG.java file.

**See Also:**

[TerResetLastMessage](#)



## TerGetOutBool

**Return the boolean value after calling certain methods.**

```
boolean TerGetOutBool(param)
```

```
String param;
```

```
// The name of the parameter for which to return the boolean value. The value  
to pass for this method is specific to individual methods, such as
```

## TerGetSectInfo.

**Description:** Certain methods save a boolean value to a boolean parameter before returning. This boolean value can be retrieved by using this method right after calling the previous method. Example:

```
if (TerGetSectInfo(LineNo)) {  
    NewPage=tej.TerGetOutBool("StartPage") // get the StartPage value  
                                           // from the previous call to  
                                           // TerGetSectInfo method  
}
```

**Return Value:** This method returns a boolean value after a method call.



## TerGetOutColor

**Return the color value after calling certain methods.**

Color TerGetOutColor(param)

String param;

// The name of the parameter for which to return the color value. The value to  
for this method is specific to individual methods, such as TerGetTextColor.

**Description:** Certain methods save a color value to an internal parameter before returning. This color value can be retrieved using this method right after calling the previous method. Example:

```
if (TerGetTextColor(FontId)) {  
    Color TextColor=tej.TerGetOutColor("TextColor"); // get the text color value  
                                                    // from the previous call to  
                                                    // TerGetTextColor method  
}
```

**Return Value:** This method returns a color value after a method call.



## TerGetOutInt

**Return the integer value after calling certain methods.**

int TerGetOutInt(param)

String param; // The name of the parameter for which to return the integer value. The value to pass for this method is specific to individual methods, such as TerGetLineInfo.

**Description:** Certain methods save an integer value to an internal parameter before returning. This integer value can be retrieved using this method right after calling the previous method. Example:

```
if (TerGetLineInfo(LineNo)) {  
    int ParaId=tej.TerGetOutInt("ParaId"); // get the paragraph id value  
                                           // from the previous call to  
                                           // TerGetLineInfo method  
}
```

**Return Value:** This method returns an integer value after a method call.



## TerGetOutMediaTray

**Return the MediaTray object after calling certain methods.**

MediaTray TerGetOutMediaTray(param)

String param; // The name of the parameter for which to return the MediaTray object. The value to pass for this method is specific to individual methods.

**Description:** Certain methods save a MediaTray object to an internal parameter before returning. This MediaTray can be retrieved using this method right after calling the previous method. Example:

```
MediaTray tray=tej.TerGetOutMediaTray("FirstPageBin");
```

**Return Value:** This method returns the MediaTray object after a method call.



## TerGetOutIntArray

**Return the integer array after calling certain methods.**

int[] TerGetOutIntArray(param)

String param; // The name of the parameter for which to return the integer



array. The value to pass for this method is specific to individual methods, such as TerGetLine.

**Description:** Certain methods save an integer array to an internal parameter before returning. This integer array can be retrieved using this method right after calling the previous method. Example:

```
if (TerGetLine(LineNo)>0) {  
    int[] fonts=tej.TerGetOutIntArray("FontArray"); // get the  
                                                    // integer array from the previous call  
to  
                                                    // TerGetLine method  
}
```

**Return Value:** This method returns the integer array after a method call.



## TerGetOutStr

**Return the string value after calling certain methods.**

String TerGetOutStr(param)

String param; // The name of the parameter for which to return the string value. The value to pass for this method is specific to individual methods, such as TerDocName.

**Description:** Certain methods save a string value to an internal parameter before returning. This string value can be retrieved using this method right after calling the previous method. Example:

```
if (TerDocName(true,"")) {  
    DocName=tej.TerGetOutStr("name"); // get the name output value  
                                        // from the previous call to  
                                        // TerDocName method  
}
```

**Return Value:** This method returns the String value after a method call.



## TerGetOutObject

**Return the output Object after calling certain methods.**

Object TerGetOutMediaTray(param)

String param; // The name of the parameter for which to return the Object. The value to pass for this method is specific to individual methods.

**Description:** Certain methods save an object to an internal parameter before returning. This Object can be retrieved using this method right after calling the previous method.

**Return Value:** This method returns the output Object after a method call.



## TerGetOutRect

**Return the Rectangle object after calling certain methods.**

Rectangle TerGetOutRect(param)

String param; // The name of the parameter for which to return the Rectangle object. The value to pass for this method is specific to individual methods.

**Description:** Certain methods save a Rectangle object to an internal parameter before returning. This Rectangle can be retrieved using this method right after calling the previous method. Example:

```
Rectangle rect=tej.TerGetOutRect("Rectangle");
```

**Return Value:** This method returns the Rectangle object after a method call.

See Also



[TerSetUserField](#)

## TerInsertUserField

**Insert a dynamic user field at the current cursor position.**

int TerInsertUserField( repaint)

boolean repaint; //Repaint the window after this operation

**Description:** This function inserts a dynamic user field. The editor fires an event called SetUserField when it encounters a user-field in the document as the page is being rendered. Your application can specify the content of the user field by using the

TerSetUserField function within this event.

The text length of the user field should not exceed one text line. When the user-field is placed in a header/footer area, the text length of the field should not change from one call to another.

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



## TerOr

**Return bitwise 'OR' value of two variables.**

```
int TerAnd(var1, var2)
```

```
int var1;                // First variable.
```

```
int var2;                // Second variable
```

**Description:** This function is useful in the programming environments which does not provide a built-in bitwise OR operator.

**Return Value:** The function returns the bitwise OR value of the input variables.

**See Also:**  
[TerAnd](#)



## TerResetLastMessage

**Reset the last editor message.**

```
boolean TerResetLastMessage()
```

**Description:** This function can be called before calling any other TEJ function to reset the last error message.

**Return Value:** The function returns true when successful.

**See Also:**  
[TerGetLastMessage](#)



## TerSetCtlColor

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

### **boolean TerSetCtlColor(color,repaint)**

Color color; // new background color.  
or, The color value can be passed as a Color object, or an  
String color; HTML color string such as red, "#ff0000".  
  
boolean repaint; // true to repaint the screen after this operation.

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



### **TerSetCustomMessage**

#### **Set custom message text for a message id.**

```
boolean TerSetCustomMessage(id, message);  
  
int id; // The message id to set the custom text. The message  
id constants (MSG_) are defined in the SubSystems.tc  
class. The original English version of the message text is  
available in the ter_msg.cs file.  
  
String message; // New message text for the message id.
```

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



### **TerSetLicenseKey**

#### **Set the license key for the product**

```
boolean TerSetLicenseKey(key);  
  
String key; // the license key. Your license key is e-mailed to you  
after your order is processed.
```

Description: This function should be called before creating any TE control to avoid pop-up nag screens.

```
Tej.TerSetLicenseKey("xxxxx-yyyyy-zzzzz")
```

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

**Return Value:** This function returns true if successful. Otherwise it indicates an invalid

license key.

See Also

[TerInsertUserField](#)



## TerSetUserField

**Set the text for the user field at the current cursor position.**

```
boolean TerSetUserField(text)
```

```
LPBYTE text; // The text for the user field.
```

**Description:** The editor fires an event called SetUserField when it encounters a user-field in the document as the page is being rendered. Your application can specify the content of the user field by using the TerSetUserField function within this event.

The text length of the user field should not exceed one text line. When the user-field is placed in a header/footer area, the text length of the field should not change from one call to another.

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



## TerUpdateDynField

**Update the dynamic fields such as page number, page-count.**

```
boolean TerUpdateDynField(repaint)
```

```
boolean repaint; // Set to TRUE to refresh the screen after the operation.
```

**Return Value:** The function returns the previous status of the speed key



## Control Properties:

This control includes a collection of design-time and run-time properties.

Although Java does not employ the term 'property', we use this term here to maintain the conceptual continuity with the Win32 and .NET versions of this control with which Tej control maintenance a certain level of compatibility

Also, the Teju applet included with this product does use the concept of property. With each property description below, you will also find the equivalent Teju class method to set the design-time or run-time properties.

### Design-time properties:

These properties must be set before the control window is created.

property name	Teju class method name (not applicable to the HTML/Javascript version)	
<b>Word Wrap:</b>	<b>setWordWrap(boolean)</b>	Turn on word wrap. (Default: True)
<b>Print View:</b>	<b>setPrintView(boolean)</b>	Edit document in the Print View mode. In this mode the lines are wrapped as they would be wrapped when printed to the selected printer (see 'Editing Mode' chapter). (Default: True)
<b>Page Mode:</b>	<b>setPageMode(boolean)</b>	Edit document one page at a time. This mode is useful when editing the documents containing multiple columns. (Default: True)
<b>FittedView:</b>	<b>setFittedView(boolean)</b>	Special case of the page mode in which the text wraps to the window width and the soft page breaks are not displayed. (Default: False)
<b>Vertical Scroll:</b>	<b>setVertScrollBar(boolean)</b>	Enable the vertical scroll bar. (Default: True)
<b>Horizontal Scroll:</b>	<b>setHorzScrollBar(boolean)</b>	Enable the horizontal scroll bar. (Default: False)
<b>Show Status Bar:</b>	<b>setShowStatusBar(boolean)</b>	Show status bar indicating the cursor position. (Default: False)
<b>Show Ruler:</b>	<b>setShowRuler(boolean)</b>	Show the ruler with tab stops and indentation indicators. (Default: False)
<b>Show Toolbar:</b>	<b>setShowToolBar(boolean)</b>	Enables tool bar. (Default: False)
<b>Border Margin:</b>	<b>setBorderMargin(boolean)</b>	Reserves a think blank area around the text box. (Default: True)
<b>Read Only:</b>	<b>setReadOnlyMode(boolean)</b>	The editor displays the text, but modifications are not allowed.
<b>TejuKey</b> (Not applicable to the HTML/Javascript version)	<b>setTejuKey(String)</b>	The Teju license key. <i>Your license key is e-mailed to you after your order is processed.</i>

<b>HtmlAddOnKey</b>	<b>setHtmlAddOnKey(String)</b>	HTML Add-on license key if HTML Add-on is installed. <i>Your license key for HTML Add-on is e-mailed to you after your order for HTML Add-on is processed.</i>
<b>SpellTimeKey</b>	<b>setSpellTikeKey(String)</b>	SpellTime license key if SpellTime is installed. <i>Your license key for SpellTime is e-mailed to you after your order for SpellTlme is processed.</i>
<b>DictPath</b>	<b>setDictPath(String)</b>	The directory where the SpellTime dictionary files are copied.
<b>UseWindow</b>	<b>setUseWindow(boolean)</b>	Set this property to False (window-less operation) when using the control within an ASP.NET server application. (Default: True)
<b>InServer</b> (Not applicable to the HTML/Javascript version)	<b>setInServer(boolean)</b>	Hosted in a server application (Default: False). Set this property to True when using the control within an ASP.NET server application.
<b>FireEvents</b> (Not applicable to the HTML/Javascript version)	<i>None</i>	<p>This property can be set for a Tejw applet to specify the name of the events to fire. The event names must be separated by the comma character.</p> <p>Example:</p> <pre>&lt;param name="FireEvents" value="ControlCreated,Preprocess"&gt;</pre> <p>The corresponding definitions for the events must exist in the html page:</p> <pre>&lt;script language="javascript"&gt; function ControlCreated() { ... any initialization code here.. }  function Preprocess(ActionType, ActionId) { if (ActionType = tej.ID_COMMAND and ActionId = tej.ID_OPEN) { .... .... } }</pre>

</script>

Please refer to the [Control Events](#) topic for the list of available events

### Other properties:

#### Data

#### setData(String)

Description: Use this property to assign or retrieve text from the control. Usage:

control.Data = String or

String = control.Data

Example:

```
control.Data = "This is a test data"
```

#### Command

#### SetCommand(int)

Description: This property is used to invoke the menu commands. The menu is not accessible when the TEJ editor is used as a control. This property allows you to access the menu commands indirectly.

*This is only a run-time properties. This method should be used only after the control is fully created.*

Usage:

```
control.Command=Command_id
```

Example:

```
tej.command = ID_PASTE
```

or

```
tej.Command(ID_PASTE)
```

*HTML/Javascript Note: The HTML/Javascript version uses the tc. prefix to access the program constants. Example: tc.ID\_PASTE .*

The command id can be one of the following:

ID\_ACCEPT\_ALL\_CHANGES

Accept all modifications

ID\_ACCEPT\_CHANGE

Accept current modification

ID\_AUTO\_SPELL

Invoke automatic spell checking. (SpellTime required for this feature)

ID\_BACK\_TAB

Enter a reverse tab

ID\_BK\_COLOR

Set background color

ID\_BKND\_PICT

Set background picture



ID_BLOCK_COPY	Copy a highlighted block
ID_BLOCK_MOVE	Move a highlighted block
ID_BORDER_MARGIN	Toggle border margin around the text box.
ID_BOLD_ON	Set bold on
ID_BOX_ON	Enable boxed option
ID_BULLET	Enable bullet option
ID_CAPS_ON	All capital letters
ID_CENTER	Center the paragraph
ID_CHAR_NORMAL	Reset the character styles
ID_CHAR_SCALEX	Expand or shrink characters horizontally
ID_CHAR_SPACE	Character spacing
ID_CHAR_STYLE	Character style
ID_COL_BREAK	Insert a column break
ID_COLOR	Choose colors
ID_CREATE_FIRST_FTR	Create the first page footer
ID_CREATE_FIRST_HDR	Create the first page header
ID_CREATE_LIST	Create a list table item
ID_CREATE_LIST_OR	Create a list override table item
ID_COPY	Copy text to clipboard
ID_CTRL_DOWN	Position at the first column of the next line.
ID_CTRL_TAB	Enter a tab character within a table
ID_CTRL_UP	Position at the first column of the previous line.
ID_CUT	Cut text to clipboard
ID_DEL	Delete the current character

ID_DEL_NEXT_WORD	Delete next word
ID_DEL_PREV_WORD	Delete the previous word.
ID_DELETE_FIRST_FTR	Delete the first page footer
ID_DELETE_FIRST_HDR	Delete the first page header
ID_DOC_RTL	Set the default right-to-left property for the document
ID_DOUBLE_SPACE	Double space the paragraph lines
ID_DOWN	Arrow down
ID_EDIT_DOB	Edit drawing object
ID_EDIT_ENOTE	Toggle editing of endnote
ID_EDIT_FNOTE	Toggle editing of footnote
ID_EDIT_HDR_FTR	Edit header/footers
ID_EDIT_INPUT_FIELD	Edit an input field
ID_EDIT_LIST	Edit the list table items
ID_EDIT_LIST_OR	Edit the list override table items
ID_EDIT_LIST_LEVEL	Edit list level properties
ID_EDIT_OLE	Edit OLE object
ID_EDIT_PICT	Edit the picture size
ID_EDIT_STYLE	Edit the style
ID_EMBED_PICT	Insert embedded picture
ID_FILE_BEGIN	Position at the beginning of the file
ID_FILE_END	Position at the end of the file
ID_FONTS	Choose fonts and point sizes
ID_FRAME_ROTATE_TEXT	Rotate the frame text
ID_FRAME_YBASE	Vertical base position

ID_HANGING_INDENT	Create or increment hanging indentation
ID_TER_HELP	Show the help window
ID_HIDDEN_ON	Set the hidden attribute on
ID_HIDE_CHANGES	Suppress the track-change display effects for text insertion and deletion.
ID_HIGHLIGHT_LINE	Highlight the line block
ID_HIGHLIGHT_TEXT	Apply text highlight color.
ID_HLINK_ON	Set the hyperlink style
ID_INLINE_IME	Inline Ime
ID_INS_AFT	Insert a line after the current line
ID_INS_BEF	Insert a line before the current line
ID_INSERT	Toggle the insert mode
ID_INSERT_BOOKMARK	Insert, delete or position at a bookmark
ID_INSERT_CHECKBOX	Insert a checkbox field.
ID_INSERT_COMBBOX	Insert a combo-box field.
ID_INSERT_DATA_FIELD	Insert a data field name and text.
ID_INSERT_DATE_TIME	Insert date/time field.
ID_INSERT_DRAW_OBJECT	Insert a drawing object
ID_INSERT_ENOTE	Insert an endnote
ID_INSERT_FNOTE	Insert a footnote
ID_INSERT_HLINK	Insert hyperlink
ID_INSERT_HYPH	Insert an optional hyphen
ID_INSERT_INPUT_FIELD	Insert an input field
ID_INSERT_NBDASH	Insert a non-breaking dash character
ID_INSERT_NBSPACE	Insert a non-breaking spacing character

ID_INSERT_PAGE_COUNT	Insert the total page count string
ID_INSERT_PAGE_NUMBER	Insert page number string
ID_INSERT_PARA_FRAME	Insert a text frame
ID_INSERT_TOC	Insert table of contents
ID_ITALIC_ON	Set italic on
ID_JOIN_LINE	Append the next line to the current line
ID_JUMP	Jump to a line number
ID_JUSTIFY	Justify paragraph on both margins
ID_LEFT	Arrow left
ID_LEFT_JUSTIFY	Left align the paragraph
ID_LEFT_INDENT	Create or increment left indentation
ID_LEFT_INDENT_DEC	Decrement left indentation.
ID_LINE_BEGIN	Position at the beginning of the line
ID_LINE_END	Position at the end of the line
ID_LINK_PICT	Insert link picture
ID_NEW	Begin a new file
ID_NEXT_CHANGE	Position at the next modified text
ID_NEXT_WORD	Position at the next word
ID_OPEN	Open an existing file
ID_PAGE_BREAK	Insert a page break
ID_PAGE_BREAK_BEFORE	Insert page break before the paragraph
ID_PAGE_OPTIONS	Set the page options
ID_PAINT_FORMAT	Apply current formatting to the selected text.
ID_PARA_BK_COLOR	Assign background color to paragraph

ID_PARA_BORDER	Create paragraph borders and shading
ID_PARA_KEEP	Paragraph keep together
ID_PARA_KEEP_NEXT	Paragraph keep with next
ID_PARA_LIST	Assign list numbering to the paragraph
ID_PARA_NORMAL	Reset the paragraph attributes
ID_PARA_NBR	Assign numbering to the paragraph
ID_PARA_RTL	Set the right-to-left property for the paragraph
ID_PARA_SPACING	Assign spacing to the paragraph
ID_PARA_STYLE	Assign style to the paragraph
ID_PASTE	Paste text from the clipboard
ID_PASTE_SPEC	Paste special clipboard formats
ID_PASTE_TEXT	Paste in the plain text format
ID_PGDN	Page down
ID_PGUP	Page up
ID_PICT_FROM_FILE	Import a bitmap from a disk file
ID_PREV_CHANGE	Position at the previous modified text
ID_PREV_WORD	Position at the previous word
ID_PRINT	Print text
ID_PRINT_OPTIONS	Set the print parameters
ID_PRINT_PREVIEW	Print preview
ID_PROTECT_FORM	Toggle the 'form protection' mode. This mode allows the user to input data into the input fields.
ID_PROTECT_ON	Set character protection on
ID_PROTECTION_LOCK	Toggle the protection lock for the document
ID_QUIT	Quit the editing session

ID_REDO	Reverse the previous undo operation
ID_REJECT_ALL_CHANGES	Reject all changes
ID_REJECT_CHANGE	Reject current change.
ID_REPAGINATE	Repaginate now
ID_REPLACE	Replace text
ID_RETURN	Process the <Enter> key
ID_RIGHT	Arrow right
ID_RIGHT_INDENT	Create or increment right indentation
ID_RIGHT_JUSTIFY	Right justify the paragraph
ID_RULER	Toggle ruler display
ID_SAVE	Save the current file
ID_SAVEAS	Save data to another file name
ID_SCAPS_ON	Small capital letters
ID_SEARCH	Search a text string
ID_SEARCH_BACK	Search for the previous text string
ID_SEARCH_FOR	Search for the next text string
ID_SECT_BREAK	Insert a section break
ID_SECT_OPTIONS	Set the section parameters
ID_SECT_RTL	Set the default right-to-left property for the section
ID_SELECT_ALL	Select the entire document
ID_SHOW_FIELD_NAMES	Show the field names
ID_SHOW_HIDDEN	Show the hidden characters
ID_SHOW_HYPERLINK_CUROSR	Toggle the display of hyperlink cursor
ID_SHOW_PAGE_BORDER	Show page borders in PageMode

ID_SHOW_PAGE_LAYOUT	Show page layout in PageMode
ID_SHOW_PARA_MARK	Toggle the paragraph marker character display
ID_SNAP_TO_GRID	Snap to grid
ID_SPELL	Invoke a spell checking session (SpellTime required for this feature) (not available for TE for HTML/Javascript)
ID_STATUS_RIBBON	Toggle the status ribbon
ID_STRIKE_ON	Set strike style on
ID_SUBSCR_ON	Set subscript on
ID_SUPSCR_ON	Set superscript on
ID_TAB	Insert a tab
ID_TAB_CLEAR	Clear a tab stop position for a paragraph
ID_TAB_CLEAR_ALL	Clear all tab stop positions for a paragraph
ID_TAB_SET	Set tab positions for a paragraph
ID_TABLE_CELL_BORDER	Edit table cell border width
ID_TABLE_CELL_BORDER_COLOR	Edit table cell border color
ID_TABLE_CELL_COLOR	Set table cell background color
ID_TABLE_CELL_SHADE	Set table cell shading
ID_TABLE_CELL_VALIGN	Set the vertical alignment for the text inside a table cell.
ID_TABLE_CELL_VTEXT	Set text rotation for a table cell text.
ID_TABLE_HDR_ROW	Toggle the 'header' attribute of the current table row
ID_TABLE_INSERT	Insert new table
ID_TABLE_CELL_WIDTH	Modify table cell width
ID_TABLE_DEL_CELLS	Delete table cells
ID_TABLE_INSERT_COL	Insert new table column

ID_TABLE_INSERT_ROW	Insert new table row
ID_TABLE_MERGE_CELLS	Merge table cells
ID_TABLE_ROW_HEIGHT	Position row height
ID_TABLE_ROW_KEEP	Keep the table row together in one page.
ID_TABLE_ROW_POS	Position table rows
ID_TABLE_ROW_RTL	Set the right-to-left property for the table
ID_TABLE_SEL_COL	select the current table column
ID_TABLE_SHOW_GRID	Show table grid lines
ID_TABLE_SPLIT_CELL	Split current table cell horizontally.
ID_TRACK_CHANGES	Toggle tracking of text modification
ID_ULINE_ON	Set underline on
ID_ULINED_ON	Set double underline attribute on
ID_UNDO	Undo previous edit
ID_UP	Arrow up
ID_USER1 to ID_USER9	Unused command ids to be used with new toolbar icons. It is generally used to implement functionality not provided by the existing command ids. For example, you can use ID_USER1 to implement PDF output icon. You would intercept ID_USER1 using the PreProcess event and execute your code there. Then call the TerIgnoreCommand method to tell the editor to ignore this command.
ID_VIEW_HDR_FTR	Show page header/footers
ID_VRULER	Toggle the display of vertical ruler
ID_WATERMARK	Let the user select a watermark picture
ID_WIDOW_ORPHAN	Enable widow/orphan control for a paragraph.
ID_ZOOM	Enable Zoom





## Control Events

Java Version: The simplest method to capture the events from Tej control is to use the TejAdapter class. Please refer to [Creating an Editor Window](#) topic an example of using this class.

```
public class TejAdapter implements TejListener {
    public void OpenFile(TejEvent e)
    public void Modified(TejEvent ev)
    public void Action(TejEvent ev)
    public void Preprocess(TejEvent ev)
    public void UpdateStatusBar(TejEvent ev)
    public void UpdateToolbar(TejEvent ev)
    public void Hypertext(TejEvent ev)
    public boolean MergeData(TejEvent ev)
    public void PageSizeChanging(TejEvent ev)
    public void Closing(TejEvent ev)
    public void PageCount(TejEvent ev)
    public void PostPaint(TejEvent ev)
    public void SpellWordReplaced(TejEvent ev) // reserved for future use
    public boolean DrawControl(TejEvent ev) // reserved for future used
    public boolean DoDNS(TejEvent ev) // reserved for future use
    public void SetUserField(TejEvent ev)
    public void ReplaceLastWord(TejEvent ev)
    public void TextDrag(TejEvent ev)
    public boolean TextDrop(TejEvent ev)
    public Component CreateEmbeddedControl(TejEvent ev)
    public void PreHtmlAddon(TejEvent ev) // reserved for future use
    public void ControlCreated(TejEvent ev)
}
```

As you would notice, these methods use the TejEvent class to pass the parameters. The description for each event includes the relevant TejEvent fields populated for the event.

**HTML/Javascript version:** The HTML/Javascript version also supports most of these events. The event must be caught within the ControlCreated function during initialization using the TerSetEvent function. Unlike the Java version, there is no TejEvent input parameter. Instead, the event parameters are passed to your script directly. Example:

```
te1.TerSetEvent("Preprocess",PreprocessEvent); // an example of catching the preprocess event
te1.TerSetEvent("Hypertext",Hypertext);
function PreprocessEvent(obj,ActionType,ActionId) {
```

```
//console.log("Preprocess, object id: ",obj.cvs.id,"type",ActionType,"id: ",ActionId);
}
```

```
function Hypertext(obj,HText,HCode) {
if (HCode.indexOf("http://")==0) window.open(HCode);
else if (HCode.indexOf("www.")==0) window.open("http://"+HCode);
else alert("Hypertext: "+HText+"\nCode/url: "+HCode);
}
```

The description for these methods follows.



## Action

**This event is sent after an user initiated action is completed.**

```
void Action(TejEvent ev)
{
    // example: Do something when the open toolbar button is clicked
    if (ev.ActionType = tej.ACTION_COMMAND
        && ev.ActionId = tej.ID_OPEN) {
        ...
    }
}
```

The relevant TejEvent fields:

```
int ActionType: Action Type
int ActionId: Action Id
```

Please refer to the the Preprocess event for the description of the ActionType and ActionId parameters.



## ControlCreated

**This event is sent after the control is fully created and the initial file/data is read into the control.**

```
void ControlCreated(TejEvent ev)
{
    ....
}
```

The relevant TejEvent fields: None

This event can be used to do the initial setup tasks such as adding or removing toolbar icons, showing the page borders, etc.



## Hypertext

**This event is sent after the user clicks on a hyperlink.**

```
void Hypertext(TejEvent ev)
{
    ....
}
```

The relevant TejEvent fields:

```
String code;           // hyperlink code (url)
String text;           // hyperlink text or phrase
boolean  DoubleClick; // true if mouse double clicked,
                       // otherwise single click
boolean  used;         // Your application should set
                       // this field to true if it acts
                       // upon this message
```

Please note that the hyperlink cursor must be toggled on to display the 'hand' cursor:

```
if (!tej.TerMenuSelect2(tej.ID_SHOW_HYPERLINK_CURSOR))
    tej.TerCommand(tej.ID_SHOW_HYPERLINK_CURSOR)
```

(turn-on the display of the 'hand' cursor if not already on)



## MergeData

**This event is sent to prompt your application for the data for a merge field.**

```
boolean MergeData(TejEvent ev)
{
    ....
}
```

The relevant TejEvent fields:

```
String name;    // field name
String data;    // field data to be set by your application
```

This event may be called when using the TerMergeFields function. Normally, you would provide the field name and field data array to the TerMergeFields function. This event is fired when the editor finds a field name in the document which is not included in the field-name array provide to the TerMergeFields function. Your application can then provide the data for the field within this event.

The 'name' parameter indicates the field name for which the data is sought.

You would use the 'data' parameter to pass the field data for the given field.

This event should return true if data is available for the specified field. Otherwise, it should return a false value.

Please also refer to the [Mail/Merge Support](#) chapter for further information.



## Modified

**This event is sent when the text is modified.**

```
void Modified(TejEvent ev)
{
    ....
}
```

The relevant TejEvent fields: None



## OpenFile

**This event is sent after a new file is read into the control.**

```
void OpenFile(TejEvent ev)
{
    ....
}
```

The relevant TejEvent fields:

```
String DocName: // The name of the newly opened file
```

This event can be used to update the file name on your application's title bar.



## PageCount

**This event is sent when page count changes as text is added or removed.**

```
void PageCount(TejEvent ev)
{

}

}
```

The relevant TejEvent fields: None



## PageSizeChanging

**This event is sent before TE adjusts the page size.**

```
void PageSizeChanging(TejEvent ev)
{
    ....
}
```

The relevant TejEvent fields:

```
int NewPageSize;
```

This event is fired only when the TFLAG5\_VARIABLE\_PAGE\_SIZE flag is set. When this flag is set, TE calculates the new page size to contain the entire content of the control. Then the control fires this event to allow your application to override or modified the suggested page size. The page size is provided by the 'NewPageSize' parameter in the twips unit. You can set the 'NewPageSize' parameter to 0 to disable the current page adjustment. You can also set it to another value to make the page bigger than the

suggested size. However, you can not set this parameter to a lower value because TE needs to display the entire content on one page.



## PostPaint

**This event is sent after TE draw the text area.**

```
void PostPaint(TejEvent ev)
{
    ....
}
```

The relevant TejEvent fields:

```
Graphics gr; // The 'gr' parameter holds the Graphics object
              associated with the editor window.
```



## Preprocess

**This event is sent before a user initiated action is processed.**

```
void Preprocess(TejEvent ev)
{
    // example: Do something when the open toolbar button is
    clicked
    if (ev.ActionType = tej.ACTION_COMMAND
        && ev.ActionId = tej.ID_OPEN) {
        ...
    }
}
```

The relevant TejEvent fields:

```
int ActionType: Action Type
int ActionId: Action Id
```

The ActionType parameter can be one of the following:

<i>ACTION_COMMAND:</i>	This action indicates any of the menu or the accelerator key generated commands. The actual command id is given by the ActionId argument. For a list of command ids, please refer to the 'command' property in the <a href="#">Control Properties</a> chapter.
<i>ACTION_VSCROLL:</i>	This action message is sent when the vertical scroll bar is clicked. The ActionId argument for this message identifies the actual scrollbar operation and is given by the SB_xxxxx SDK constants.
<i>ACTION_HSCROLL</i>	This action message is sent when the horizontal scroll bar is clicked. The ActionId argument for this message identifies the actual scrollbar operation and is given by the SB_xxxxx SDK constants.
<i>ACTION_CHAR:</i>	This action message is sent when the editor processes a WM_CHAR message. The ActionId argument for this message indicates the virtual key code for the key.
<i>ACTION_LBUTTONDOWN</i> <i>N</i>	Left mouse button down. The ActionId contains the x and y mouse position in the pixel units. The x position is given by the low 16 bits and the y position is given by the high 16 bits.
<i>ACTION_RBUTTONDOWN</i> <i>N:</i>	Right mouse button down. The ActionId parameter holds the mouse position as described for the ACTION_LBUTTONDOWN message.
<i>ACTION_LBUTTONUP:</i>	Left mouse button up. The ActionId parameter holds the mouse position as described for the ACTION_LBUTTONDOWN message.
<i>ACTION_RBUTTONUP:</i>	Right mouse button up. The ActionId parameter holds the mouse position as described for the ACTION_LBUTTONDOWN message.
<i>ACTION_LBUTTONDBLCLICK:</i>	Left mouse double click. The ActionId parameter holds the mouse position as described for the ACTION_LBUTTONDOWN message.
<i>ACTION_RBUTTONDBLCLICK:</i>	Right mouse double click. The ActionId parameter holds the mouse position as described for the ACTION_LBUTTONDOWN message.
<i>ACTION_MOUSEMOVE:</i>	Mouse move. The ActionId contains the x and y mouse position in the pixel units. The x position is given by the low 16 bits and the y position is given by the high 16 bits.
<i>ACTION_SIZE:</i>	Window being resized.

<code>ACTION_SETFOCUS:</code>	Tej window receiving focus.
<code>ACTION_KILLFOCUS:</code>	Tej window losing focus.
<code>ACTION_QUERYENDSESSION</code>	This message is sent before asking the user to end the editing session.
<code>ACTION_STYLE</code>	This message is sent when the user selects a different style from the style combo-box. The ActionId specifies the selected style id.

ActionId: This value is specific to the action type as described above.

See Also: [TerIgnoreCommand](#)



## ReplaceLastWord

**This event is fired during keyboard text entry after the user completes a word. It lets you provide a replacement word for the word just entered.**

```
void ReplaceLastWord(TejEvent ev)
{
    // an example of replace the last word entered
    ev.replace=true;
    ev.NewText=" ";

    if (ev.LastWord.equals("dog")) ev.NewText="Golden Retriever";
    else if (ev.LastWord.equals("cat")) ev.NewText="Himalayan tiger";
    else ev.replace=false;
}
```

The relevant TejEvent fields:

```
String LastWord: // The 'LastWord' parameter indicates the word
                 just completed.
boolean replace; // The 'replace' parameter should be set to
                 true if you are providing a replace text.
String NewText;  // The 'NewText' parameter should be set to
                 the replacement text.
```





## SetUserField

**This event is fired when a user-field is encountered when the text is being rendered.**

```
void SetUserField(TejEvent ev)
{
    ....
}
```

The relevant TejEvent fields:

```
int PageNo;    // page number (zero based)
int TextPos;   // text-position of the field
```

This event uses two arguments. The PageNo argument specifies the page number (zero based) where the user-field was encountered. The TextPos argument provides the absolute text position of the field. The absolute text position can be converted into the line/column position using the TerAbsToRowCol method.

Typically your application would set the text value of the current user field when this event is encountered. The text for the field is set using the TerSetUserField method.



## SpellWordReplaced

**This event is fired when the spell-checker replaces a misspelled word.**

```
void SpellWordReplaced(object Sender,int CharPos,
                       String OldWord, String NewWord)
{
    ....
}
```

The relevant TejEvent fields:

```
int CharPos;    // The character position of the replaced word.
String OldWord; // The original word.
String NewWord; // The replacement word.
```

The 'CharPos' parameter indicates the character position of the replaced word. You can use the TerAbsToRowCol method to convert the character position to a line/column value.



## UpdateStatusbar

**This event is sent when an external status bar needs to be repainted.**

```
void UpdateStatusbar(TejEvent ev)
{
    ....
}
```

The relevant TejEvent fields: None



## UpdateToolbar

**This event is sent when an external toolbar needs to be repainted.**

```
void UpdateToolbar(TejEvent ev)
{
    ....
}
```

The relevant TejEvent fields: None



## CreateEmbeddedControl

**This event is sent when an embedded control is encountered during RTF input.**

```
Component CreateEmbeddedControl(TejEvent ev)
{
    JComponent ctl=null;
    if (ev.ClassName.equals("TextBox")) {
```

```

        JTextField tb=new JTextField();
        if (id==1) tb.setText("Name");
        else      tb.setText("Address");
        return (Component)tb;
    }
    return (Component)ctl;
}

```

The relevant TejEvent fields:

```

String ClassName;
int id;

```

Your application would use the event to recreate the embedded control which was originally inserted into the input RTF file using the TerInsertControl method. The value of the 'id' parameter is the same the 'id' parameter value specified when calling the TerInsertControl method.

Tej control already includes a built-in support for these components: JTextArea, JCheckBox, JComboBox, JButton, JRadioButton, and JLabel. This event is needed to support additional component classes.



## Tejw Applet

This package includes a Tejw applet which can be inserted into your web page. The operating system must support standard Java to be able to use this applet.

If you are using ASP.NET, then please follow the [ASP.NET Interface](#) topic for the relevant instructions.

You can also insert the applet into the html page by simply using the <applet> tag. Here is an example:

```

<applet id="tej" name="tej" archive="tej31.jar" code="Tejw.class" width=1000
height=750 MAYSCRIPT
        style="border-width:0px; border-style:none;
height:416px;width:755px;Z-INDEX: 2;
        LEFT: 14px; POSITION: absolute; TOP: 47px">
<param name="FireEvents" value="ControlCreated">
<Param name="WordWrap" value=True>
<Param name="PageMode" value=True>
<Param name="PrintViewMode" value=True>
<Param name="FittedView" value=False>
<Param name="ShowStatusBar" value=True>
<Param name="ShowRuler" value=True>
<Param name="ShowVertRuler" value=True>
<Param name="ShowToolBar" value=True>
<Param name="BorderMargin" value=False>

```

```
<Param name="RTFOutput" value=True>
<Param name="ReadOnlyMode" value=False>
<Param name="VertScrollBar" value=True>
<Param name="HorzScrollBar" value=True>
<Param name="TejKey" value=""> // your license key goes here
<Param name="Data" value="{\rtf \i Please type here... \i0}">
<Param name="InWebPage" value=True>
</applet>
```

Please refer to the included WebEdit.zip file for a demo using the applet tag in an html page.



## ASP.NET Interface

The ASP.NET wrapper allows you to drop a TE Edit Control into your ASP.NET page.

The ASP.NET wrapper dll for HTML/Javascript is named WebTeh, whereas the ASP.NET wrapper for the Java version is named WebTej. Please refer to one of the following two topics depending upon the platform version of the editor control you are using.

### In This Chapter

[TE Edit Control for HTML/Javascript](#)

[TE Edit Control for Java - Applet](#)



## TE Edit Control for HTML/Javascript

This section describes the ASP.NET wrapper for TE Edit Control for HTML/Javascript

If you are using TE Edit Control for Java (applet), then please refer the subsequent chapter.

The ASP.NET wrapper for TE Edit Control for HTML/Javascript is contained in a dll called WebTeh.dll.

WebTeh.dll is a wrapper to TE Edit Control component to be used in a Web application.

You can drop a WebTeh object as you design a web page in Visual Studio. As the page is loaded on the client machine, the resulting html page will include an 'canvas' tag to contain the edit control.

This chapter describes the WebTeh properties, and events.

### In This Chapter

[Demo Application](#)

[WebTeh Control](#)

[WebTeh Control Properties](#)

[Teh Control Events](#)



## Demo Application

DmoTeh application included with the product shows the usage of the WebTeh and TE Edit for HTML/Javascript component.

To run the demo application, please unzip the dmo\_web\_teh.zip file to a folder under the c:\inetpub\wwwroot directory. Let us say for example that you would unzip this file in the following directory:

c:\inetpub\wwwroot\demo

You will see the following files unzipped:

WebTeh.dll	Editor web control. This control generates the 'canvas' tag to host the Teh control within the html page generated by ASP.NET
Default.aspx.cs	Source for the demo
Defaultj.aspx	The aspx file to be loaded by the client machine.
AssemblyInfo.cs	The assembly file for the demo project.

Now copy the WebTeh.dll file from the demo directory to the c:\inetpub\wwwroot\demo\bin directory.

Also copy the tejh25.js, ter\_hlp.pdf, ter\_tlb.gif and wait.gif files to the c:\inetpub\wwwroot\demo directory (not the bin directory) These files include the code and the support files for the editor. These files gets downloaded by the browser as the client page shows the editor control.

Now you can access the demo from the client machine:

`http://www.myserver.com/demo`

You can also access the demo application from your own machine:

`http://localhost/demo`



## WebTeh Control

WebTeh.dll control is a simple drop-in control to be used within an ASP.NET web application.

This dll is included in the demo\_web\_teh.zip file. Please unzip this file and copy the WebTeh.dll file to the c:\inetpub\wwwroot\bin directory.

To use the WebTeh.dll in a web application, create a new web application (or open an existing one) and right click on the 'Web Forms' tab in the Toolbox. Now select the 'Add/Remove new items...' and add the WebTeh.dll control. Now the control will be available for selection in the toolbox.

Now you can select and drop an instances of the WebTeh control on your web application page. The control properties should be set in the PageLoaded method of your ASP.NET app:

```
protected void Page_Load(object sender, System.EventArgs e) {  
    // set position and other properties for the control  
    WebTeh1.Left = 25;  
    WebTeh1.Top = 40;  
    WebTeh1.Width = 800;  
    WebTeh1.Height = 410;
```

The WebTeh control generates the 'canvas' tag to host the TE edit control in the web page. Here is an example of the 'canvas' tag generated by this control:

```
<script src="terh31.js"></script>  
  
<canvas id="WebTeh1" width="800" height="410" tabindent="0"  
    style="position:absolute; left:25px; top:40px; z-index:0; border:1px  
solid; "  
    data-word-wrap="true"  
    data-page-mode="true"  
    data-print-view-mode="true"  
    data-fitted-view="false"  
    data-show-menu="true"  
    data-show-status-bar="true"  
    data-show-ruler="true"  
    data-show-vert-ruler="true"  
    data-show-tool-bar="true"  
    data-border-margin="true"  
    data-rtf-output="true"  
    data-read-only-mode="false"  
    data-vert-scroll-bar="true"  
    data-horz-scroll-bar="true"  
    data-spell-time-dict-path=""  
    data-data="{\rtf\i Please type here...}">
```

Sorry, this browser does not support HTML 5.

</canvas>

The terh31.js, ter\_hlp.pdf, ter\_tlb.gif and wait.gif files must be copied to the application directory for the page to be loaded properly.

The License Key delivered must be copied to a file and placed in your project folder as described in your order delivery e-mail.



## WebTeh Control Properties

There are some common properties between the Teh and WebTeh controls. There properties include: **WordWrap**, **PageMode**, **PrintViewMode**, **FittedView**, **ShowStatusBar**, **BorderMargin**, **ShowRuler**, **ShowToolBar**, **ReadOnlyMode**, **VertScrollBar**, **HorzScrollBar**, and **Data**. Please refer to the [Control Properties](#) section for a description of these common properties.

The control properties should be set in the PageLoaded method of your ASP.NET app:

```
protected void Page_Load(object sender, System.EventArgs e) {  
    // set position and other properties for the control  
    WebTeh1.Left = 25;  
    WebTeh1.Top = 40;  
    WebTeh1.Width = 800;  
    WebTeh1.Height = 410;
```



## Teh Control Events

WebTeh control does not support any server-side events. However, WebTeh automatically generates a call to the ControlCreated function so as to provide an opportunity to perform further initialization. So your web page

must include a Control Created function to catch this event. You would add code for further initialization and code to catch other events within this function. Here is an example:

```
<script>
    var tej=null;    // a global variable to hold the Tej control object
    function ControlCreated(WebTej) // this gets called when the edit control is
// created and displayed
    {
        tej = WebTej;
        console.log("Page is loaded!");

        // Example of adding or removing toolbar icons
        tej.TerAddToolbarIcon(0,0,tc.ID_SUPSCR_ON,
                               "images/icon1.bmp","superscript");
        tej.TerAddToolbarIcon(0,0,tc.ID_SUBSCR_ON,
                               "images/icon2.bmp","subscript");
        tej.TerHideToolbarIcon(tc.TLB_HELP,true); // remove this icon

        tej.TerRecreateToolbar(true);

        // Example of disabling a speedkey
        tej.TerEnableSpeedKey(tc.ID_CHAR_STYLE,0); // disable the original
                                                    // Alt+1 key command

        // Catch other event
        tej1.TerSetEvent("Preprocess",PreprocessEvent); // an example of
                                                         // catching the preprocess event
        tej1.TerSetEvent("Hypertext",Hypertext);
    }
</script>
```

In addition, you can catch all the events support by the Teh Control.

The embedded Teh control fires the regular events within the WebPage. A full description of the available event names and corresponding input parameter can be found in the [Control Events](#) topic.

Please see the example above to catch the Preprocess and Hypertext event. The first parameter specifies the name of the event to catch, and the second parameter specify the function to call. So, you would define two separate function to resolve the references to the PreprocessEvent and Hypertext event. Example:

```
<script>
    function PreprocessEvent(obj, ActionType, ActionId) {
        console.log("Preprocess, object id: ",obj.cvs.id,"type",
                    ActionType,"id: ",ActionId);
    }
</script>
```



```
    }  
  
    function Hypertext(obj,HText,HCode) {  
        if (HCode.indexOf("http://")==0) window.open(HCode);  
        else if (HCode.indexOf("www.")==0) window.open("http://"+HCode);  
        else alert("Hypertext: "+HText+"\nCode/url: "+HCode);  
    }  
</script>
```

In the examples above, WebTej refers the 'canvas' hosting the Teh control.  
Please refer to the included Default.aspx and terh.htm files for a complete example.



## TE Edit Control for Java - Applet

This section describes the ASP.NET wrapper for TE Edit Control for Java.

If you are using TE Edit Control for HTML/Javascript, then please refer the subsequent chapter.

The ASP.NET wrapper for TE Edit Control for Java is contained in a dll called WebTej.dll.

WebTej.dll is an applet wrapper to Tej component to be used in a Web application. The operating system must support standard Java to be able to use this applet.

You can drop a WebTej object as you design a web page in Visual Studio. As the page is loaded on the client machine, the resulting html page will include an 'applet' tag which refers to the Tej control.

This chapter describes the WebTej properties, and events.

### In This Chapter

[Demo Application](#)

[WebTej Control](#)

[WebTej Control Properties](#)

[WebTej Control Events](#)



## Demo Application

DmoTej application included with the product shows the usage of the WebTej and Tej component.

To run the demo application, please unzip the dmo\_web\_tej.zip file to a folder under the c:\inetpub\wwwroot directory. Let us say for example that you would unzip this file in the following directory:

c:\inetpub\wwwroot\demo

You will see the following files unzipped:

WebTej.dll	Editor web control. This control generates the 'object' tag to host the Tej control within the html page generated by ASP.NET
DmoTej.aspx.cs	Source for the demo
DmoTej.aspx	The aspx file to be loaded by the client machine.
AssemblyInfo.cs	The assembly file for the demo project.

Now copy the WebTej.dll file from the demo directory to the c:\inetpub\wwwroot\demo\bin directory.

Also copy the tej31.jar and tej31.jar.pack.gz files to the c:\inetpub\wwwroot\demo directory (not the bin directory)

Now you can access the demo from the client machine:

`http://www.myserver.com/demo/DmoTej.aspx`

You can also access the demo application from your own machine:

`http://localhost/demo/DmoTej.aspx`



## WebTej Control

WebTej.dll control is a simple drop-in control to be used within an ASP.NET web application.

This dll is included in the dmo\_web\_tej.zip file. Please unzip this file and copy the WebTej.dll file to the c:\inetpub\wwwroot\bin directory.

To use the WebTej.dll in a web application, create a new web application (or open an existing one) and right click on the 'Web Forms' tab in the Toolbox. Now select the 'Add/Remove new items...' and add the WebTej.dll control. Now the control will be available for selection in the toolbox.

Now you can select and drop an instances of the WebTej control on your web application page. Right click on the control to set the control [properties](#) as desired.

The WebTej control generates the 'applet' tag to host the Tej edit control in the web page. Here is an example of the 'applet' tag generated by this control:

```
<applet name="WebTej1" id="WebTej1" code="Tejw.class" id="WebTej1"
Text="WebTej" style="height:712px;width:1284px;z-index: 101; left: 0px;
```

```
position: absolute;
        top: 0px">

        <Param name="cache_archive" value="tej31.jar">
        <Param name="java_arguments" value="-Djnlp.packEnabled=true">

        <Param name="FireEvents" value="">
        <Param name="WordWrap" value=True>
        <Param name="PageMode" value=True>
        <Param name="PrintViewMode" value=True>
        <Param name="FittedView" value=False>
        <Param name="ShowMenu" value=True>
        <Param name="ShowStatusBar" value=True>

        <Param name="ShowRuler" value=True>
        <Param name="ShowVertRuler" value=True>
        <Param name="ShowToolBar" value=True>
        <Param name="BorderMargin" value=True>
        <Param name="RTFOutput" value=True>
        <Param name="ReadOnlyMode" value=False>
        <Param name="VertScrollBar" value=True>
        <Param name="HorzScrollBar" value=True>
        <Param name="TejKey" value="">

        <Param name="HtmlAddOnKey" value="">
        <Param name="SpellTimeKey" value="">
        <Param name="SpellTimeDictPath" value="">
        <Param name="Data" value="">
        <Param name="InWeb" value=True>

</applet>
```

The tej31.jar file must be copied to the application directory for the page to be loaded properly. Your license keys must be assigned to the TejKey property using the property box. This control requires a Server license (not a Desktop license).



## WebTej Control Properties

There are some common properties between the Tej and WebTej controls. There properties include: **WordWrap**, **PageMode**, **PrintViewMode**, **FittedView**, **ShowStatusBar**, **BorderMargin**, **ShowRuler**, **ShowToolBar**, **ReadOnlyMode**, **VertScrollBar**, **HorzScrollBar**, **TejKey**, and **Data**. Please refer to the [Tej Control Properties](#) section for a description of these common properties.

In addition to the common properties listed above, the WebTej control supports the following new properties:

<b>ServerDataXfer</b>	When this property is TRUE, the WebTej control generates a hidden field to update the 'Data' property when the form is submitted. Set this property to False if you do not need to retrieve the updated control data.
<b>HtnKey</b>	The license key for the HTML Add-on product, if HTML Add-on is installed. (reserved for future use)
<b>StnKey</b>	The license key for the SpellTime product, if SpellTime is installed. (reserved for future use)
<b>DictPath</b>	The name of the folder on the client machine which contains the SpellTime dictionary files. (reserved for future use)



## WebTej Control Events

WebTej control does not support any server-side events.

The embedded Tej control fires the regular events within the WebPage. The names of the events to handle is specified using the FireEvents property. You can set this property using the property-box in VisualStudio. A full description of the available event names and corresponding input parameter can be found in the [Control Events](#) topic.

For example, if you set the FireEvents property to "ControlCreated,Preprocess", the following property statement would be created in the page generated by IIS:

```
<Param name="FireEvents" value="ControlCreated,Preprocess">
```



```
}  
  
//-->  
</SCRIPT>
```

In the examples above, WebTej1 refers to the 'object' hosting the Tej control.  
Please refer to the included DmoTej.aspx file for a complete example.



## Mail/Merge Support

The product supports two kinds of mail-merge fields.

### **RTF type mail/merge fields:**

First, the RTF type of mail-merge is created using the TerInsertField function, and populated using the TerChangeField function for each field in the document. The TerLocateField function is used to locate the RTF type of fields in the document. The advantage of using the RTF type of mail-merge field is that the field-name is stored separately from the field-data. Therefore, when the data is applied to the field using the TerChangeField field, you can still use the field-name to locate the field again. Click here for the detail description of these [Mail-merge](#) functions.

### **Double-underline type mail/merge fields:**

The second type of fields are created by simply typing the field name and then underlining it. Please refer to the merge.rtf file for an example. The advantage of using this type of fields is that you can merge all fields in one call to the TerMergeFields function. The second advantage is that you are able to do merge and print as one step without opening a TE window using the TerMergePrint or TerMergePrintVB functions. In this topic we will explore this simple type of mail merge operation.

The simple mail/merge method consists of two components. The first component involves the user who creates a document containing the data field names. The second component is your application which calls the mail/merge print API to print a mail/merge document replacing the field names with field data.

**Creating a Mail/Merge Document:** A mail merge document is very similar to an ordinary document. To insert a field name in a document, do the following:

- Input the field name using the keyboard.
- Highlight the text for the field name.
- Select the 'double underline' option from the font menu to apply the double underline style to the field name. The 'double underline' style is used to indicate the field names.

**Printing a Mail/Merge Document:** A mail merge document must be printed within your program's control. (The 'Print' option in the 'File' menu can not be used to print a

mail/merge document).

Your program initiates a mail/merge printing by using the 'TerMergePrint' function. Please refer to the 'Application Interface Functions' chapter for the complete description of this function.

Your application passes the print specification to the 'TerMergePrint' function using the 'TejPrint' class variable. The 'TerMergePrint' function is called for each record that you wish to merge and print in the document. The following two member variables within the 'TejPrint' class are used for supplying data for the field names:

*MergeFields*: This field specifies the variable to a list of mail merge field names. Each field name must be separated by a '|' character. The list must be terminated by a null character. If you do not wish to merge field data, set this field to null.

*MergeData*: This field specifies the variable to a list of mail merge data strings. Each data String must be separated by a '|' character. The number of data elements in the 'MergeData' array MUST be the same as the number of elements in the 'MergeFields' array. The list must be terminated by a null character. If you do not wish to merge field data, set this field to null.

Example:

```
MergeFields="name|address|city|st|zip";  
MergeData="Jim|139 Main St|Springfield|MA|02371"
```

The 'TerMergePrint' function scans the document to extract the field names. If a field name is found in the 'MergeFields' array, the corresponding String in the 'MergeData' is used to replace the field name with the data String in the document.

If the field name is not found in the 'MergeFields' array, the 'TerMergePrint' function sends a MergeData event to your application.

Example of using the MergeData event:

```
protected boolean MergeData(object Sender, String name,  
                             out String data)  
{  
    data="";  
    if (name=="date") {  
        data=DateTime.Today.Date.ToString();  
        return true;  
    }  
    return false;  
}
```

This example returns today's date as a data String for the 'date' field. Please also refer to the [MergeData](#) event for further information.



## Hyperlink Hooks

The editor provides the hooks to implement hyperlink facility.

**Activation** When the user double clicks on the text formatted with the double underline attribute, the editor sends a Hypertext event to the parent window. The hyperlink text format can be changed from double underline to any format of your choice by using the following code.

```
GetTerFields(out field)
```

```
field.LinkStyle = style-constant
```

```
field.LinkColor = color
```

```
field.LinkDbfClick = set to false to activate the hyperlink on a single mouse click.
```

```
tej1.SetTerFields(field)
```

For a list of character styles, please refer to the SetTerCharStyle function. A special style-constant called HLINK is also available. This style does not have any visible attribute, but it allows the hyperlink text to have any mix of fonts and colors.

### Event

Example:

```
protected void Hypertext(object Sender,
                          TejHyperlink link)
{
    MessageBox.Show(link.text, link.code,
                   MessageBoxButtons.OK);
    link.used=true;
}
```

The 'TejHyperlink' class is defined as following:

```
public class TejHyperlink {
    public String code;        // hyperlink code
    public String text;       // hyperlink text
    public boolean DoubleClick;// TRUE if mouse double
clicked,
                                otherwise single click
    public boolean used;      // the host sets it to true if
it
                                acts upon this message
};
```

The 'text' member variable stores the text formatted with the double underline attribute.

The 'code' member variable stores the 'hidden' text found *immediately* before the link



text.

Your application should set the 'used' variable to a true value if it processes this message. Otherwise it should return a false value.



## Editing Modes

The TEJ editor offer these four editing modes:

### Text Mode:

The text mode is initiated when the editor is called with word wrapping turned *off*. This mode is most suitable for editing the text files such as computer programs and batch files. In this mode, the lines are not wrapped automatically. This mode does not offer the paragraph formatting features.

### Word Wrap Mode:

This mode is initiated when the routine is called with the word wrapping turned *on*. In this mode, the text in a window is automatically formatted to wrap at the end of the line. Therefore the complete line of text is always visible regardless of the window width. A special character 'ParaChar' is used to delimit a paragraph. This character is not displayed on the screen. Additionally, you have an option of suppressing this character when the file is written out to the disk.

This mode also allows the character and paragraph formatting features.

### Print View Mode:

This mode is initiated when the editor is called with both the Word Wrap and the Print View flags turned on. In this mode, the text lines are wrapped as they would be wrapped when printed to the selected printer. The horizontal scrolling is automatically provided when the text goes beyond the current width of the window. This mode offers all the features of the Word Wrap Mode. In addition, it provide automatic repagination. This mode also allows for sections with multiple columns.

### Page Mode:

This mode is initiated when the editor is called with both the Word Wrap and the Page Mode flags turned on. As in the Print View mode, the text lines are wrapped as they would be wrapped when printed to the selected printer. In this mode, however, the editor displays one page at a time. This mode is most useful for the documents containing multiple columns, as the columns are displayed side by side. In addition, this mode provides all the features of the Print View mode

### FittedView:

This is a special case of the page mode in which the text wraps to the window width and the soft page breaks are not displayed



## Text Editor Commands

This chapter describes the editor commands by menu groups.

### In This Chapter

[How To Scroll Through The Text](#)

[File and Print Commands](#)

[Line Edit Commands](#)

[Block Edit Commands](#)

[Clipboard Commands](#)

[Picture Commands](#)

[Character Formatting Commands](#)

[Paragraph Formatting Commands](#)

[Paragraph Spacing, Borders and Shading](#)

[Tab Support](#)

[Page Break and Repagination](#)

[Page Header/Footer, Bookmark and Footnote Commands](#)

[Table Commands](#)

[Section and Columns](#)

[Stylesheet and Table-of-contents](#)

[Text/Picture Frame and Drawing Objects](#)

[View Options](#)

[Navigation Commands](#)

[Search/Replace Commands](#)

[Highlighting Commands](#)



## How To Scroll Through The Text

### Keyboard:

Use *Up*, *Down*, *Left* and *Right* arrow keys to scroll up or down a line, or left or right one character.

. Hit the *Home* key to position at the beginning of the current line.

Hit the *End* key to position at the end of the current line.

Hit *Ctrl-PgUp* to position at the beginning of a file.

Hit *Ctrl-PgDn* to position at the end of a file.

Hit *PgUp* to display the previous page.

Hit *PgDn* to display the next page.

Hit *Ctrl* - *Left* arrow key to position on the next word.

Hit *Ctrl* - *Right* arrow key to position on the previous word.

Hit *Ctrl* - *Up* arrow key to position at the first column of the current line (if not already on the first column) or at the first column of the previous line.

Hit *Ctrl* - *Down* arrow key to position at the first column of the next line.

Hit the *F10* key and type in the line number to jump to. This function is also available from the Navigation menu.

#### **Mouse:**

You can click mouse on the vertical and horizontal scroll bar to accomplish various scrolling function. These functions are available only if the horizontal or the vertical bar has been enabled by the startup parameters:

**Vertical Scroll Bar:** Click the mouse on the arrows on either end to scroll the screen up or down by one line. Click the mouse above the elevator to scroll the screen up by one page. Similarly, click the mouse below the elevator to scroll the screen down by one page. You may also drag the elevator to any position in the bar. As the elevator is dragged, the editor will scroll the screen up or down accordingly to maintain the correct cursor position.

**Horizontal Scroll Bar:** Click the mouse on the arrows on either end to scroll the screen left or right by one line. Click the mouse on either side of the elevator to scroll the screen left or right by 1/2 screen. You may also drag the elevator to any position in the bar. As the elevator is dragged, the editor will scroll the screen left or right accordingly to maintain the correct cursor position.



## **File and Print Commands**

**New File** This function is used to clear the existing text from the edit window and start an empty, unnamed document. The user is prompted to save any modification to the previous document.

**Open File** This function is used to clear the exiting text from the edit window and open a new document. The user is prompted to save any modification to the previous document.

**Save File** Use this selection to save the text to the current file name. If a file is not yet specified, the editor will prompt you for a file name. If a file with the same name already exists on the disk, the editor will save the previous file with a backup extension (.TE).

If the I/O is conducted through a buffer rather than a disk file, the editor creates a new buffer with the updated text.

You can invoke this function by hitting the F3 function key (or select the option

from the menu).

**Save File As..** This selection is similar to *Save File*. In addition, it allows you to specify a new file name for saving the text.

This option is not available when the I/O is conducted through a buffer rather than a disk file.

You can invoke this function by hitting the Shift F3 function keys together (or select the option from the menu)

**Exit** Use this function to exit from the editor session. If the current file is modified, you will have an option to save the modifications.

You can invoke this function by hitting the Ctrl F3 function keys together (or select the option from the menu).

**Print** Use this option to print the contents of the current file. You may also choose to print only the selected part of the file. To print a block of text, the desired text must be highlighted before invoking the print function. This command supports these highlighted blocks:

Line Block

Character Block

The print function will print on a default printer selected from the Windows' control panel. You can alter the *printer setup* or *Page Layout* prior to invoking the print option.

You can invoke the printing function by hitting the F4 function key (or select the option from the menu). The editor will display a dialog box where you can select the scope of the printing.

**Page Layout** Use this option before selecting the *Print* option to specify the page layout. You can specify margin (left, right, top and bottom) in inches.

You can invoke this function by hitting the Ctrl F4 function keys together (or select the option from the menu).

**Printer Setup** This option invokes a printer specific dialog box for the default printer (the default printer selection is made from the control panel of Windows) You select the parameters from a set of printer specific options. These options include page size, page orientation, resolution, fonts, etc.

You can invoke this function by hitting the Shift F4 function keys together (or select the option from the menu).

**Print Preview** This option is used to preview the document before printing. The editor displays up to 2 pages at a time. You can scroll to a different page by using the PgUp/PgDn or the scroll bar.

By default the preview rectangle is sized to fit the current window. However, you can use the zoom option to enlarge or shrink the preview rectangle as you wish.



## Line Edit Commands

- Insert After Current Line** In the *text mode* this function creates a blank line after the current line. Hit the F9 function key to insert a line after the current line.
- Insert Before Current Line** In the *text mode* this function creates a blank line before the current line. Hit the Ctrl F5 keys together to insert a line before the current line.
- Delete Line** Use this function to delete the current line. The remaining lines will be scrolled up by one line. Hit the Shift F9 keys together to delete the current line.
- Join Lines** In the *text mode* this function joins the next line at the end of the current line. Hit the Alt J keys together to invoke this function.
- Split Line** In the *text mode* this function splits the current line at the current cursor position. Hit the Alt S keys together to invoke this function.



## Block Edit Commands

- Copy a Line Block** Use this command to **copy** a highlighted block of text lines from one location to another. This command provides a short alternative to using clipboard copy/paste functions.
- Highlight the lines of text to be copied, move the caret to the target location and hit Alt C (or select the option from the menu). This function does not delete the original block.
- Move a Line Block** Use this command to **move** a highlighted block of text lines from one location to another. This command provides a short alternative to using clipboard cut/paste functions.
- Highlight a block of text to be moved, move the caret to the target location and hit Alt M (or select the option from the menu). This function deletes the original block.
- Undo Previous Edit** The editor remembers your last edit command. You can use this function to undo the last edit command.
- You can invoke this function by hitting the Shift F8 keys together (or select the option from the menu). The editor will display a dialog box containing the information about the edit command to be undone. The dialog box displays the line number, column position, type of undo (delete/insert/edit) and the contents of the undo buffer. You may modify the target line number or column position. Confirm the operation by clicking on the OK button.
- This undo feature is not available for column block edits, block move and replace String commands.

**Redo Previous Undo** This command reverses the previous undo operation.



## Clipboard Commands

### Cut/Copy Text To Clipboard

Use this command to **cut or copy** a highlighted block of text to the clipboard. This function also copies the associated formatting information using the RTF format and the native TEJ format.

Highlight a block of text to be copied to the clipboard and hit the Ctrl+X (cut) or Ctrl+C (copy) keys, or select the option from the menu.

### Paste Text From Clipboard

Use this command to paste the contents of the clipboard at the current caret location. The formatting information, if available, is also copied.

You can invoke this function by hitting the Ctrl+V keys together (or select the option from the menu).

### Paste Special Objects

This function displays the clipboard data in a number of available formats:

#### Native Object Format

If available this is the first format in the list box. The data in this format can be later edited (by double clicking the object) using the *original* application. This data can be *embedded* into your application by using the Paste option, or you can create a *link* to the original file by using the Paste Link option.

#### Formatted Text

This is one of the text formats. This option offers the most suitable format if the data is pasted by another text output application as the font and formatting attributes are reproduced accurately.

#### Unformatted Text

This is another text format. This option pastes the text without retaining the formatting information.

#### Picture Format

The data is available in the Picture format. This object can be later edited (by double clicking the object) using the Microsofts MS Draw application. This format is preferred over the bitmap and the device independent bitmap formats.

#### Device Independent and regular bitmap formats

The data is available in the bitmap formats. The object can be later edited (by double clicking the object) using the Microsofts MS Draw application. The editor converts these formats into the Picture format before calling the drawing application.

:



## Picture Commands

- Embed Picture** Use this command to embed a picture bitmap or metafile from a disk file at the current caret location. The embedded picture is saved within the document.
- Link Picture** Use this command to link a picture bitmap or metafile to the document. The linked picture appears at the current caret location. A linked picture data is not saved with the document, only its name is store with the document.
- Edit Picture** Use this command to change the width and height of a picture located at the current caret position. The width and height is specified in inches. This function also allows you to align (top, bottom, or middle) the picture relative to the base line of the text.
- Drag/Drop Function** This is a method of inserting a file object into the text directly. To insert a file, open the Windows File Manager and locate the file to be inserted. Now click the mouse and keep the mouse button depressed as you move the mouse cursor to the editor window. Release the mouse button at the location where the object should be inserted. The editor shows an icon to indicate the inserted object. You can edit this object by double clicked at the icon.
- The object inserted using this method makes use of Microsofts Packager application to tie the file with the application that originally created it.
- Please note that a documented problem with the original Packager application may create errors during this function. Install the corrected version of the PACKAGER.EXE program for proper functioning.
- Background Picture** This option, available from the 'Other' menu, is used to set a background picture for the text. The background picture occupies the entire text area. The picture file can be a Windows' bitmap (.BMP) or Metafile (.WMF).



## Character Formatting Commands

**Character Styles** The following character style commands are available:

<b>Command</b>	<b>&amp;nbsp;</b>	<b>Keystroke</b>	<b>&amp;nbsp;</b>
Normal		Alt	0
Bold Formatting		Ctrl	B
Underlining		Ctrl	U
Italic		Ctrl	I
Superscript		Alt	4
Subscript		Alt	5
Strike		Alt	6

Character style options allows you to apply one or more style formats to the current character or to all characters in a highlighted block of text.

To apply a format to the current character, simply hit the appropriate keystroke (or select the option from the menu). To apply this format on a block of characters, highlight a block using the Line Block or Character Block options. Now, hit the applicable keystroke, or select the option from the menu.

When you type in on the keyboard, the new characters automatically assume all the formatting characteristics of the preceding character.

TEJ allows multiple formats for a character. To apply more than one format, repeat the procedure described in the previous paragraphs.

To reset all character formats, highlight the characters and select the 'Normal' option from the menu, or hit the Alt 0 keystroke.

### Fonts

Use this option to change the font typeface and point size of the current character or of all characters in a highlighted block of text.

If you wish to change the font for a highlighted block of text, highlight the block using the Line or Character highlight function. If you wish to change the font of a single character, simply position the cursor on that character. Now select the font option from the menu or hit the Alt F10 keys together. A dialog box will appear that shows the list of typefaces and point sizes to select from. Make the desired selection now.

### Colors

Use this selection to change the text color of the current character or of all characters in a highlighted block of text.

If you wish to change the color of a highlighted block of text, highlight the block using the Line or Character highlight function. If you wish to change the color of a single character, simply position the cursor on that character. Now select the color option from the menu. A dialog box will appear that shows the color selection. Make the desired selection now.

### Hidden Text

The text formatted with this attribute are treated as hidden text. Normally the hidden text, as the name implies, does not appear on the screen or printer. However you can display the hidden text by selecting the 'Show Hidden Text' option from the 'View' menu.

### Protected Text

The text formatted with this attribute are protected from the editing changes. The protected text appear with a light shade in the window. This function is available only when the 'protection lock' is turned off. The 'protection lock' can be turned off by using an option from the 'Other' menu.



## Paragraph Formatting Commands

### Reset Paragraph Format

Use this selection to reset all paragraph formats for the current paragraph or for all lines in a highlighted block of text.



To reset the paragraph formats for the current paragraph, simply hit the Alt P keys together (or select the option from the menu). To reset the formats for a block of lines, highlight a block and hit the Alt P Keys together (or select the option from the menu).

### **Paragraph Centering**

Use this selection to center all lines in the current paragraph or all lines in a highlighted block of text.

To center the current paragraph, simply hit the Alt 8 keys together (or select the option from the menu). To center a block of lines, highlight a block of text and hit the Alt 8 Keys together (or select the option from the menu).

### **Paragraph Right Justification**

Use this selection to right justify all lines in the current paragraph or all lines in a highlighted block of text.

To right justify the current paragraph, simply hit the Alt 9 keys together (or select the option from the menu). To right justify a block of lines, highlight a block of text and hit the Alt 9 Keys together (or select the option from the menu).

### **Paragraph Justification**

Use this selection to justify the text on both left and right margins.

To justify the current paragraph, simply select the option from the paragraph menu. To justify a block of lines, highlight a block of text and then select this option from the menu.

### **Paragraph Double Spacing**

Use this selection to double space all lines in the current paragraph or all lines in a highlighted block of text. A double spaced paragraph has a blank line between each text line.

To double space the current paragraph, simply hit the Alt O keys together (or select the option from the menu). To double space a block of lines, highlight a block of text and hit the Alt O Keys together (or select the option from the menu)

### **Paragraph Indentation (Left)**

Use this selection to create a left indentation for all lines in the current paragraph or for all lines in a highlighted block of text. The successive use of this option increases the amount of left indentation.

To apply the left indentation to the current paragraph, simply hit the Alt L keys together (or select the option from the menu). To apply the left indentation to a block of lines, highlight a block of text and hit the Alt L Keys together (or select the option from the menu).

To create the left indentation using the mouse, click the left mouse button on the indentation symbol on the lower left end of the ruler. While the mouse button is depressed, drag the mouse to the desired location and release the mouse button. The indentation created using this method is applicable to every line in the paragraph except the first line.

### **Paragraph Indentation (Right)**

Use this selection to create a right indentation for all lines in the current paragraph or for all lines in a highlighted block of text. The successive use of this option increases the amount of right indentation.

To apply the right indentation to the current paragraph, simply hit the Alt

R keys together (or select the option from the menu). To apply the right indentation to a block of lines, highlight a block of text and hit the Alt R Keys together (or select the option from the menu).

To create the right indentation using the mouse, click the left mouse button on the indentation symbol on the lower right end of the ruler. While the mouse button is depressed, drag the mouse to the desired location and release the mouse button.

### **Paragraph Hanging Indentation**

This option is similar to paragraph left indentation, except that the indentation is not applied to the first line of the paragraph.

To apply the hanging indentation to the current paragraph, simply hit the Alt T keys together (or select the option from the menu). To apply the left indentation to a block of lines, highlight a block of text and hit the Alt T Keys together (or select the option from the menu).

To create the hanging indentation using the mouse, click the left mouse button on the indentation symbol on the upper left end of the ruler. While the mouse button is depressed, drag the mouse to the desired location and release the mouse button.

### **Paragraph Keep Together**

When this attribute is turned on for a paragraph, the editor attempts to keep all lines within the paragraph on the same page.

### **Paragraph Keep with Next**

When this attribute is turned on for a paragraph, the editor attempts to keep the last line of the current paragraph and the first line of the next paragraph on the same page.

### **Widow/Orphan Control**

When this attribute is turned on for a paragraph, the editor attempts to avoid widow/orphan paragraphs. An 'orphan' paragraph results when the last line of the paragraph lies on the next page. A 'widow' paragraph results when the first line of the paragraph lies on the previous page



## **Paragraph Spacing, Borders and Shading**

This functionality is provided by two options in the paragraph menu, one to set the Border and Shading parameters and the other to set the spacing parameters for a paragraph.

The '**Paragraph Spacing**' menu option allows you to set the space before and after the paragraph. You can also specify the minimum space between the paragraph lines. All space parameters are specified in points.

The '**Border and Shading**' option in the paragraph menu allows you to create the paragraph borders and set the shading amount for the paragraph. You can draw all four sides of the border, or you can draw only the selected sides. Additional two options allow you to select a thick and double lined border.

When two or more contiguous paragraphs have identical paragraph formatting parameters, a single border is drawn to enclose all such contiguous paragraphs.

The top line of the border is placed beneath the top of the first line. The bottom line of the border is placed above the bottom of the last line. Create a blank line at the top and

bottom if you need additional clearance at the top or bottom. The left line of the border is placed before the left indentation for the paragraph. Therefore, the left side may not be visible for the paragraph with no left indentation. The right line of the border is placed after the right indentation. Therefore, the right side may not be visible for the paragraph where the right margin extends up to or beyond the width of the window.



## Tab Support

TE Editor supports left, right, center, and decimal tab stops. The tab stops are very useful for creating columns and tables. A paragraph can have as many as 20 tab positions.

The 'left' tab stop begins the text following a tab character at the next tab position. To create a left tab stop, click the left mouse button at the specified location on the ruler. The left tab stop is indicated on the ruler by an arrow with a tail toward the right.

The 'right' tab stop aligns the text at the current tab stop such that the text ends at the tab marker. To create a right tab stop, click the right mouse button at the specified location on the ruler. The right tab stop is indicated on the ruler by an arrow with a tail toward the left.

The 'center' tab stop centers the text at the current tab position. To create a center tab stop, hold the shift key and click the left mouse button at the specified location on the ruler. The center tab stop is indicated on the ruler by a straight arrow.

The 'decimal' tab stop aligns the text at the decimal point. To create a decimal tab stop, hold the shift key and click the right mouse button at the specified location on the ruler. The decimal tab stop is indicated on the ruler by a dot under a straight arrow.

The tab stops can also be created by using the 'Set Tab' selection from the 'Paragraph' menu. This option allows you to specify the tab position, tab type (left, right, center, or decimal) and tab leader (dot, hyphen, underline, or none).

To move a tab position using the mouse, simply click the left mouse button on the tab symbol on the ruler. While the mouse button is depressed, drag the mouse to the desired location and release the mouse button.

To clear a tab position, simply click at the desired tab marker, or select the option from the menu. You can also clear all tab stops for the selected text by selecting 'Clear All Tabs' option from the menu.

The 'Snap To Grid' option in the 'Other' menu affects the movement of the tabs (and the paragraph indentation markers) on the ruler. When this option is checked, the movements of these markers are locked on to an invisible grid at an interval of 1/16 inch.

Normally, a tab command is applicable to every line of the current paragraph. However, if you highlight a block of text before initiating a tab command, the tab command is then applicable to all the lines in the highlighted block of text.



## Page Break and Repagination

A hard page break can be inserted in the document by pressing the Control and Enter keys together (or select the option from the menu: Edit->Break->Section Break). A hard page break places the text after the page break on the following page. A hard page break is indicated by a solid line in the editing window.

In the Print View editing mode, the editor also creates automatic page breaks when the text overflows a page. An automatic page break is indicated by a dotted line in the editing window. As the name implies, these page breaks are calculated automatically by the editor between the keystrokes. The repagination process is time consuming. Sometimes there may not be enough time for a large document to complete the repagination between the edits. Therefore, the menu also provides an option to provide complete repagination on demand.

**Inserting Page Number** The 'Page Number' selection from the 'Insert' menu allows you to insert the page number into the document. The page number String is inserted at the current cursor position. This String is displayed using a gray color.

**Inserting Page Count** The 'Page Count' selection from the 'Insert' menu allows you to insert the total number of pages into the document. The page count String is inserted at the current cursor position. This String is displayed using a gray color.

**Show Page Border** When option is turned on, the editor displays the borders around the text on the screen. This option is available in the page mode only. The 'FittedView' option must be turned off.



## Page Header/Footer, Bookmark and Footnote Commands

The page header/footer functionality is available in the Page Mode only.

**Show Page Header/Footer** Normally, the editor does not show the header and footer for a page. You can use this option from the 'View' menu to display the page header and footer.

This option does not allow you to edit the text for the page header/footer. Every section in a document can have its own page header and footer. If a section does not have a page header/footer of its own, this option shows the header/footer from the preceding section for the pages in this section.

**Edit Page Header/Footer** The user can use this option to edit the text for the page header and footer. This option is available from the 'Edit' menu.

- Insert Footnote** This option allows you to insert a footnote at the current cursor location. The footnote is displayed at the bottom of the page.
- Edit Footnote Text** This option displays the footnote text in-line with the regular text. It allows you to edit the footnote text. The modified footnote is displayed at the bottom of the page.
- Insert Bookmark** This dialog box is activated from the 'Insert' menu. It allows you to place a bookmark (new or existing) at the current text location. You can also position the cursor at a specified bookmark. It also allows you to delete an existing bookmark.



## Table Commands

The table menu is available in the Page mode or Print View modes only (see Editing Modes). This menu contains the commands to create a new table or to edit table attributes.

- Insert Table** Use this option to insert a new table in the document. This option prompts the user for the initial number of rows and columns in the table. The editor initially creates the cells of equal width. The user can, however, change the cell width by dragging the cell borders using the mouse.
- In the Page Mode, the table cells are arranged by rows. In the Print View Mode, the table structure is not visible.
- Insert Table Row** Use this option to insert a new row before the current table row. The new table row has the same number of columns as the current table row.
- Merge Table Cells** Use this option to merge together the highlighted cells. The width of the resulting cells is equal to the sum of all merged cells. If the highlighted cells span more than one table row, this operation creates multiple merged cells each within its row.
- Split Table Cell** Use this option to split the current table cell into two cells of equal width. The entire text of the original cell is assigned to the first cell. The second cell is created empty.
- Delete Table Cells** Use this option to delete the selected cells from the table. A dialog box allows the user to select the cells for the deletion.
- The dialog box has three options: cells, columns, and rows. The first option selects the current cell or all the cells in the highlighted block of text. The second option selects all the cells in the current column or the columns containing the cells in the highlighted block of text. The third option selects all the cells in the current row or the rows containing the cells in the highlighted block of text.

A table is automatically deleted when all its cells are deleted.

**Table Row Position** Use this option to position the table or a selected table rows. A dialog box lets you position the table as left justified, centered, or right justified.

**Table Cell Border** Use this option to create the borders around the selected cells. A dialog box allows the user to select the cells for this operation.

The dialog box has three options: cells, columns, and rows. The first option selects the current cell or all the cells in the highlighted block of text. The second option selects all the cells in the current column or the columns containing the cells in the highlighted block of text. The third option selects all the cells in the current row or the rows containing the cells in the highlighted block of text.

The user can specify the width of each border (top, bottom, left and right). The border width should be less than the cell text margin. The cell text margin is the distance from the left edge of the cell to the beginning of the text in the cell. The border width is specified in twips (1440 twips equal to one inch).

**Table Cell Shading** Use this option to shade the selected cells. A dialog box allows the user to select the cells for this operation.

The dialog box has three options: cells, columns, and rows. The first option selects the current cell or all the cells in the highlighted block of text. The second option selects all the cells in the current column or the columns containing the cells in the highlighted block of text. The third option selects all the cells in the current row or the rows containing the cells in the highlighted block of text.

The shading is specified in terms of the shading percentage. A value of 0 indicates a white background, where as the value of 100 indicates a black background. A value between 0 and 100 indicates the level of shading.

**Show Table Grid Lines** Use this option to enable or disable the display of the table grid lines. The table grid lines are for display purpose only, they are not drawn when printing to a printer



## Section and Columns

The editor allows you to divide a document into multiple sections. A multiple section document is useful when a) you need to vary the page margins from one page to another and b) you need to create multiple column text.

**Creating a New Section** To create a new section, select the 'Break' submenu option from the 'Edit' menu. A section break line (double solid line) is created before the current line. The new section begins at the text following the break line.

**Editing the Section** The following section parameters can be edited:

## Parameters

*Number of columns and column spacing.*

*Portrait or Landscape orientation.*

*Placement of the text on the next page.*

*Page Margins*

The first three parameters can be edited by selecting the 'Section Edit' option from the 'Edit' menu. The last parameter can be edited by selecting the 'Page Setup' option from the 'File' Menu.

**Deleting a section break line** To delete a section break line, simply position the cursor on the section break line and hit the <DEL> key.

**Multiple Column Editing** This option is available in the Print View and Page Modes only (See Editing Modes)

To create multiple columns for a section, select the 'Section Edit' option from the menu and specify the number of columns to create. You can also specify the space between the columns.

The text in the multiple column section wraps at the end of the column. When the text reaches the end of the page, or the end of a section, the new text is placed on the next column.

In the Print View mode, the multiple columns are not actually seen in the window. In the Page Mode, the columns are visible as they would be when the text is printed. Therefore, the Page Mode is useful when editing multiple column text.

## Column Break

Normally in a multiple column section, the text flows to the next column at the end of the current column. The column break option can be used to force the text to the next column before the current column is completely filled.

A column break can be inserted by selecting the option from the menu (Edit->Insert Break...). A column break is indicated by a line with a 'dot and dash' pattej. The text after the column break line is placed on the next column. To delete the column break line, simply position the cursor on the line and hit the <DEL> key



## Stylesheet and Table-of-contents

The editor supports the character and paragraph type stylesheet style items. The character stylesheet style constitutes a set of character formatting attributes and is applied to a character string. The paragraph stylesheet style constitutes not only a set of character formatting attributes, but also a set of paragraph formatting attributes. The paragraph style is applied to one or more paragraphs.

## Create and edit styles

A stylesheet style is created and modified using the 'Edit Style' menu option from the 'Edit' menu. This option displays a dialog box which allows you to choose between a character style or a paragraph style. You can select an existing style to modify from the list box or enter the name for the new style. Once you click the 'Ok' button, the recording of the stylesheet properties begins. You can use the ruler, toolbar, or the menu selections to modify the stylesheet items. The ruler, toolbar, and menu also reflect the currently selected properties for the stylesheet item. Please note that the paragraph properties are allowed only for the paragraph type of stylesheet item.

After you have selected the desired properties, terminate the stylesheet editing mode by either selecting the 'Edit Style' selection from the menu again or by clicking anywhere in the document. If the existing stylesheet item was modified, the document automatically reflects the updated stylesheet properties. If a new stylesheet item was created, your next step is to apply the style to the desired text by choosing the 'style' option from the 'Font' or the 'Paragraph' menu selection.

### Apply character styles

The 'style' menu selection in the 'Font' menu allows you to apply a stylesheet style to the currently highlighted character string.

### Apply paragraph styles

The 'style' menu selection in the 'Paragraph' menu allows you to apply a stylesheet style to the current paragraph. To apply a style to a range of paragraphs, highlight the paragraphs before selecting the 'style' menu option.

### Table of Contents

To insert a table of contents, first create the heading styles using the 'Edit Style' option from the 'Edit' menu. For example, if you wish to insert a three level deep table of contents, create heading styles 'heading 1', 'heading 2', and 'heading 3'. Then place the cursor at the heading lines and apply a suitable heading style using 'style' menu selection from the 'Paragraph' menu. The last step would be to position the cursor where you wish to insert the table of contents and select the 'Table of Contents' menu selection from the 'Insert' menu.

The table-of-contents are automatically updated whenever repagination occurs.



## Text/Picture Frame and Drawing Objects

A frame is a rectangular area on the page. A frame can contain both text and picture. The text outside the frame flows around the frame. A drawing object can be a text box, rectangle or a line. The drawing object overlays on top of the text

The 'Frame' or 'Drawing Object' option from the 'Insert' menu is used to embed a frame or a drawing object into the text. The new object is inserted at the current text position.

To insert text into the frame or a text box, click a mouse button inside the frame to select



the frame. Now type the text at the cursor position.

To size a frame, click a mouse button inside the frame to select the frame. Now click the left mouse button on a sizing tab and move the mouse while the mouse button is depressed. Release the mouse when done. The text inside the frame is automatically wrapped to adjust to the new width. If the new height of the frame is not enough to contain all text lines, the frame height is automatically adjusted to include all lines. If the frame contains only a picture, the picture size is automatically adjusted to fill the frame.

To move the frame, click a mouse button inside the frame to select the frame. Now move the mouse cursor just outside the frame until a plus shaped cursor appears. Click the left mouse button. While the mouse button is depressed, move the frame to the new location and release the mouse button.

To edit the base vertical position of the frame, select the 'Vertical Frame Base...' option from the edit->frame menu. The frame locked to the top of the page or the top of the margin retain their vertical position when the text is inserted before them.

To edit the border and the background of a drawing object, select the 'Edit Drawing Object' option from the edit->frame menu.

This option is available in the Page Mode only.



## View Options

This menu allows you to turn on and off the following viewing options:

### Page Mode

In this mode, the editor displays one page at a time. This mode is available when the editor is called with both the Word Wrap and the Page Mode (or the PageView flag) flags turned on. This mode is most useful for the documents containing multiple columns, as the columns are displayed side by side. In addition, this mode provides all the features of the Print View mode.

### FittedView

Special case of the page mode in which the text wraps to the window width and the soft page breaks are not displayed

### Ruler

The ruler shows tab stops and paragraph indentation marks. The ruler can also be used to create or delete tab stops

### Tool Bar

The tool bar provides a convenient method of selecting fonts, point sizes, character styles and paragraph properties. The tool bar also shows the current selection for font, point size and character styles.

### Show Status Ribbon

The status ribbon displays the current page number, line number, column number and row number. It also indicates the current insert/overtyping mode.

### Show Hidden Text

This option displays the text formatted with the hidden attribute (see

Character Formatting Options) with a dotted underline. When this option is turned off, the hidden text is not visible.

**Show Paragraph Mark** This option displays a symbol (an inverted 'P') at the end of each paragraph. This option may be useful when working with lines with many different heights

**Hyperlink Cursor** This option is used to display the hyperlink cursor when the cursor is positioned on a hypertext phrase. The hyperlink cursor is an image of a hand with a finger pointing to the text.

**Zoom** This feature allows you to compress or enlarge the display of the document text. The editor allows a zoom percentage between 25 and 200.



## Navigation Commands

**Jump** Use this function to position on a desired line number.

You can invoke this function by hitting the F10 function key (or select the option from the menu). The editor will then display a dialog box so that you can enter the line number to jump to.

See 'How to Scrolling Through the Text' section for other navigation functions.



## Search/Replace Commands

**Search a Text String** Use this function to locate a String of characters in the current file. The editor will search for the first instance of the given character string. To find the subsequent instances of the same character string, use *Search Forward* or *Search Backward* commands.

You can invoke this function by hitting the F5 function key (or select the option from the menu). The editor will display a dialog box where you enter the character String to locate. You can specify the search to be in the backward or the forward direction from the current cursor position or you can specify the search to take place from the beginning of the file. You can also force a non-casesensitive search, in which case the String is matched irrespective of the case of the letters in the string.

- Search Forward** Use this function to locate the next instance of a previously located String using the *Search Function*. If the Search Function is not yet invoked, this function will call the Search Function instead.
- You can invoke this function by hitting the Control F Keys together (or select the option from the menu).
- Search Backward** Use this function to locate the previous instance of a previously located String using the *Search Function*. If the Search Function is not yet invoked, this function will call the Search Function instead.
- You can invoke this function by hitting the Control Shift F Keys together (or select the option from the menu).
- Replace a Text String** Use this function to replace a character String with another character string.
- You can invoke this function by hitting the F6 function key (or select the option from the menu). The editor will show a dialog box where you will enter the old and new character strings. You may also choose to conduct the replace only within a selected part of the file. To choose such a block of text, the desired text must be highlighted before invoking the replace function.
- The dialog box also offers you an option to force the editor to verify each replace.



## Highlighting Commands

- Highlight a Character Block** Use this function to highlight a block of characters.
- Mouse: Position the mouse cursor on the first character of the block and depress the left button. While the left button is depressed, drag the mouse to the last character of the block and release the mouse.
- Keystroke: Position the caret on the first character of the block and press the shift key. While the shift key is pressed, use the position keys to move the caret on the last character of the block and release the shift key. Normally, you can also use any position key in combination with the Shift key to create, expand, or shrink the text selection.
- Normally, a function that utilizes a character block, also erases the highlighting. To explicitly erase the highlighting click a mouse button again or press any position key.
- Highlight a Line Block** Use this function to highlight a block of lines.
- Mouse: Position the mouse cursor at any position on the first line of the block and depress the right button. While the right button is depressed, drag the mouse to the last line of the block and release the mouse.

Keystroke: Position the caret at any position on the first line of the block and hit the F8 function key. Use the Up and Down arrow keys to position the caret on the last line and hit F8 again.

Normally, a function that utilizes a line block, also erases the highlighting. To explicitly erase the highlighting click a mouse button again or press the F8 key again.

### **Highlight a Word**

Double click any mouse button on the desired word to highlight the word