C#
Windows, .NET (.NET 1.0)
Win32, VS (VS.NET2002)
Dev
Posted : **15 Jul 2003**
Views : **139,425**

Insert Plain Text and Images into RichTextBox at Runtime

By **Khendys Gordon**.

This article describes how to programmatically insert text and images into a RichTextBox at runtime.

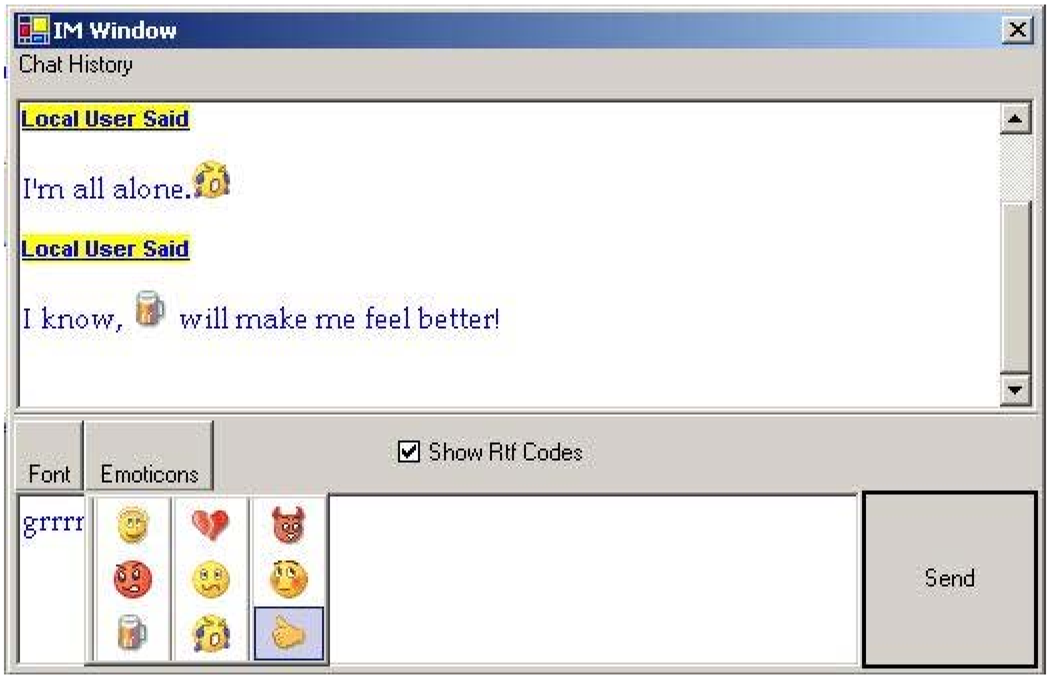
Programming books
Ultimate Combo MFC tools $449
Send to a friend

Search Articles Go! Advanced Search Sitemap

Print Broken Article? Bookmark Discuss

72 votes for this article.
Popularity: 8.67. Rating: **4.67** out of 5.

Download source - 42.1 Kb



Introduction

If you've searched the Internet for an easy way to insert images into a `RichTextBox`, chances are you've come across variations of the following solution which copies the image to the clipboard, pastes it in the target `RichTextBox`, and clears the contents of the clipboard.

```
public void InsertImage()  {
  ...
  string lstrFile = fileDialog.FileName;
  Bitmap myBitmap = new Bitmap(lstrFile);
```

```
   // Copy the bitmap to the clipboard.
   Clipboard.SetDataObject(myBitmap);
   // Get the format for the object type.
   DataFormats.Format myFormat = DataFormats.GetFormat (DataFormats.Bitmap);
   // After verifying that the data can be pasted, paste
   if(NoteBox.CanPaste(myFormat)) {
     NoteBox.Paste(myFormat);
   }
   else {
     MessageBox.Show("The data format that you attempted site" +
       " is not supportedby this control.");
   }
   ...
 }
```

This is not a good solution because it alters the clipboard without informing the user, which can be a real inconvenience. Other solutions hard-code thousands of lines of the HEX representation of images into the program, but that's not very flexible (or practical). There is also no standard way of inserting plain text into a `RichTextBox` at runtime. This article offer a solution to these problems.

The solution must:

1. Allow plain text to be programmatically inserted into or appended to the content if a `RichTextBox` at runtime.
2. Allow the font, text color, and highlight color (background color of the text) to be specified when inserting or appending plain text to the content of a `RichTextBox`.
3. Allow images to be inserted programmatically without the use of the clipboard.

The content of a `RichTextBox` can be in either plain text format or Rich Text Format. Henceforth Rich Text Format is simply as RTF.

NOTE: Converting plain text to RTF is really about appending strings to create the RTF codes. It is very simple, but one needs to be familiar with the RTF document structure and control words. In an effort not to turn the article into an RTF tutorial, the methods for inserting plain text will be discussed briefly, however for a full explanation the reader should view the source code and should read the RTF Specification v1.6.

Background

Before getting into the solution, an introduction to RTF documents and Metafiles is warranted.

RTF Documents

RTF is a structured file format that uses control words and symbols to create a file that can be used in different operating environments. When being read, the RTF control words and symbols are processed by an RTF reader which converts RTF into formatted text. This is similar to how a browser displays HTML to a user. In this case, the RTF reader is the `RichTextBox`.

The RTF Specification is a 250+ page document, so attempting to summarize it in this article would be a severe injustice to its authors. The only RTF control words that will be explained are those used when inserting an image. For a complete introduction to RTF, please read RTF Specification v1.6.

Metafiles

In the .NET Framework, the `Metafile` class is derived from the `Image` class, however metafiles are not raster images like those that can be converted to `Bitmap` objects. A raster image is composed of rectangular arrays of pixels known as bitmaps. A metafile is a vector image which contains a geometrical representation of an image in terms of drawing commands. A `Metafile` can be converted to a `Bitmap` using the .NET Framework, but a `Bitmap` cannot be converted to a `Metafile` using .NET only. However, bitmaps can be embedded within metafiles.

The .NET Framework offers support for two types of metafiles: Windows Metafile Format (WMF) and Enhanced Metafile Format (EMF). These metafiles differ in the drawing commands they support; Enhanced Metafiles support many more drawing commands than Windows Metafiles. According to Microsoft's documentation, the WMF format should not be used and is only included for backward compatibility, however this solution uses the WMF. For complete documentation on metafiles click here.

Inserting and Appending Plain Text

Insertion of RTF into a `RichTextBox` is done by assigning a string representation of an RTF document to the `RichTextBox.Rtf` property or the `RichTextBox.SelectedRtf` property. When the latter is used to insert, if there is text selected at the time of insertion, the text will be replaced. If no text is selected, the text is inserted at the location of the caret.

Appending Text

Plain text is appended to the content of the `RichTextBox` by moving the caret to the end of the RTF text in the `RichTextBox` and performing an insert.

```
/// Appends the text using the given font, text,
/// and highlight colors.  Simply
/// moves the caret to the end of the RichTextBox's text
/// and makes a call to insert.
public void AppendTextAsRtf(string _text, Font _font,
  RtfColor _textColor, RtfColor _backColor) {

  // Move carret to the end of the text
  this.Select(this.TextLength, 0);

  InsertTextAsRtf(_text, _font, _textColor, _backColor);
}
```

There are three other overloads of `AppendTextAsRtf` which all eventually call the overload above.

```
/// Appends the text using the current font, text, and highlight colors.
public void AppendTextAsRtf(string _text) {
  AppendTextAsRtf(_text, this.Font);
}

/// Appends the text using the given font, and
/// current text and highlight colors.
public void AppendTextAsRtf(string _text, Font _font) {
  AppendTextAsRtf(_text, _font, textColor);
}
```

```
/// Appends the text using the given font and text color, and the current
/// highlight color.
public void AppendTextAsRtf(string _text, Font _font, RtfColor _textColor) {
  AppendTextAsRtf(_text, _font, _textColor, highlightColor);
}
```

Inserting Text

When inserting text into a `RichTextBox`, the text must be a document in Rich Text Format. An RTF document consists of a *header* and *document* area which must conform to the RTF Specification. The header consists of, among other things, the language being used, and tables of the fonts and colors used in the document. The document area is where the actual contents of the document are stored and formatted. Upon a call to the `InsertTextAsRtf` method, an RTF header is constructed and the plain text is added to and formatted in the document area.

```
  public void InsertTextAsRtf(string _text, Font _font,
    RtfColor _textColor, RtfColor _backColor) {

    StringBuilder _rtf = new StringBuilder();

    // Append the RTF header
    _rtf.Append(RTF_HEADER);

    // Create the font table from the font passed in and append it to the
    // RTF string
    _rtf.Append(GetFontTable(_font));

    // Create the color table from the colors passed in and append it to the
    // RTF string
    _rtf.Append(GetColorTable(_textColor, _backColor));

    // Create the document area from the text to be added as RTF and append
    // it to the RTF string.
    _rtf.Append(GetDocumentArea(_text, _font));

    this.SelectedRtf = _rtf.ToString();
  }
```

There are three other overloads to `InsertTextAsRtf` which are shown below. For an in-depth look at this procedure please view the source code and refer to the RTF Specification v1.6.

```
  /// Inserts the text using the current font, text, and highlight colors.
  public void InsertTextAsRtf(string _text) {
    InsertTextAsRtf(_text, this.Font);
  }


  /// Inserts the text using the given font, and current text and highlight
  /// colors.
  public void InsertTextAsRtf(string _text, Font _font) {
    InsertTextAsRtf(_text, _font, textColor);
  }
```

```
/// Inserts the text using the given font and text color, and the current
/// highlight color.
public void InsertTextAsRtf(string _text, Font _font,
  RtfColor _textColor) {
  InsertTextAsRtf(_text, _font, _textColor, highlightColor);
}
```

Inserting an Image

When an image is pasted into a `RichTextBox` (or WordPad or Microsoft Word), the image is embedded in a Windows Metafile (WMF) and the metafile is placed in the document. The `InsertImage` method does the same thing, but without using the clipboard. According to the RTF Specification v1.6, it is possible to insert bitmaps, JPEGs, GIFs, PNGs, and Enhanced Metafiles directly into an RTF document without first embedding them in a Windows Metafile. However, while this works with Microsoft Word, if images are not embedded in a Windows Metafile, WordPad and `RichTextBox` simply ignore them.

Metafiles

A Windows Metafile is the metafile format that was originally supported on Windows 1.0 (1985). They have limited capabilities and are supported in the Windows Forms only for backward compatibility. .NET does not directly support the creation of Windows Metafiles, but it can read them. The creation of Enhanced Metafiles is supported, however, and Enhanced Metafiles can be converted to Windows Metafiles using unmanaged code. GDI+ (*gdiplus.dll*) contains a function called `EmfToWmfBits()`, which converts an Enhanced Metafile to a Windows Metafile. This function is shown below.

```
[DllImportAttribute("gdiplus.dll")]
private static extern uint GdipEmfToWmfBits (IntPtr _hEmf,
  uint _bufferSize, byte[] _buffer,
  int _mappingMode, EmfToWmfBitsFlags _flags);
```

`_hEmf` is the handle to the Enhanced Metafile being converted. `_bufferSize` is the size of the buffer used to store the converted Windows Metafile. `_buffer` is an array of bytes used to store the converted Windows Metafile. `_mappingMode` refers to the mapping mode of the image. The mapping modes define the orientation and units used to transform the image, and are provided by the Windows API. `MM_ANISOTROPIC` is used as the mapping mode in this solution. It allows both axes of the image to be changed independently. `_flags` indicate the options for converting the metafile.

The image is embedded in an Enhanced Metafile by drawing the the image onto a graphics context created from the metafile. The Enhanced Metafile is then converted to a Windows Metafile. The Enhanced Metafile is created using the constructor overload below.

```
Metafile(Stream stream, IntPtr referencedHdc);
```

This creates a new Enhanced Metafile using the device context `referencedHdc` and stores it in `stream`. A device context is a structure that contains information that controls the display of text and graphics on a particular device. Metafiles need to be associated with a device context to obtain resolution information. Every `Graphics` object can provide a handle to its device context. The device context of the

RichTextBox is obtained by making a call to its GetGraphics method, and then calling the GetHdc method of the resultant Graphics object.

```
...

// Memory stream where Metafile will be stored
_stream = new MemoryStream();

// Get a graphics context from the RichTextBox
using(_graphics = this.CreateGraphics()) {

  // Get the device context from the graphics context
  _hdc = _graphics.GetHdc();

  // Create a new Enhanced Metafile from the device context
  _metaFile = new Metafile(_stream, _hdc);

  // Release the device context
  _graphics.ReleaseHdc(_hdc);
}

...
```

A Graphics context is now created from the metafile and the image is drawn (embedded) in the file.

```
...

// Get a graphics context from the Enhanced Metafile
using(_graphics = Graphics.FromImage(_metaFile)) {

  // Draw the image on the Enhanced Metafile
  _graphics.DrawImage(_image, new Rectangle(0, 0,
    _image.Width, _image.Height));

}

...
```

Calling EmfToWmfBits with a null buffer parameter returns the size of the buffer necessary to store the Windows Metafile. This is used to create an array large enough to hold the Windows Metafile.

```
...

// Get number of bytes
uint _bufferSize = GdipEmfToWmfBits(_hEmf, 0, null,
  MM_ANISOTROPIC, EmfToWmfBitsFlags.EmfToWmfBitsFlagsDefault);

// Create an array to hold the file
byte[] _buffer = new byte[_bufferSize];

...
```

Calling EmfToWmfBits with an instantiated buffer copies the metafile into the buffer and returns the number of bytes copied.

```
...

// Get the file
uint _convertedSize = GdipEmfToWmfBits(_hEmf, _bufferSize,
  _buffer, MM_ANISOTROPIC,
  EmfToWmfBitsFlags.EmfToWmfBitsFlagsDefault);

...
```

A HEX representation of the image is created from the array and is now ready to be inserted into the RichTextBox.

```
...

// Append the bits to the RTF string
for(int i = 0; i < _buffer.Length; ++i) {
  _rtf.Append(String.Format("{0:X2}", _buffer[i]));
}

return _rtf.ToString();

...
```

RTF Picture Destination

The minimum control words used to define a picture or image in an RTF document are "{\pict\wmetafile8\picw[N]\pich[N]\picwgoal[N]\pichgoal[N] [BYTES]}" where ...

\pict	- The starting picture or image tag
\wmetafile[N]	- Indicates that the image type is a Windows Metafile. [N] = 8 specifies that the metafile's axes can be sized independently.
\picw[N] and \pich[N]	- Define the size of the image, where[N] is in units of hundreths of millimeters (0.01)mm.
\picwgoal[N] and \pichgoal[N]	- Define the target size of the image, where [N] is in units of twips.
[BYTES]	- The HEX representation of the image.

The horizontal and vertical resolutions at which the ExRichTextBox is being displayed are necessary for the above calculations to be made. These values are obtained in the default constructor of ExRichTextBox from a Graphics object and stored as xDpi and yDpi respectively. (On most systems, both these values are 96 Dpi, but why assume?)

The metafile's dimensions in (0.01)mm are calculated using the following conversion units and formula. (The example below explains how to find the current width, but the same formula is used to find the height by substituting height and vertical resolution for width and horizontal resolution respectively.)

1 Inch = 2.54 cm
1 Inch = 25.4 mm
1 Inch = 2540 (0.01)mm

[N]	= current width of the metafile in hundredths of millimeters (0.01mm)
	= Image Width in Inches * Number of (0.01mm) per inch
	= (Image Width in Pixels / Graphics Context's Horizontal Resolution) * 2540
	= (Image Width in Pixels / Graphics.DpiX) * 2540

```
...

// Calculate the current width of the image in (0.01)mm
int picw = (int)Math.Round((_image.Width / xDpi) * HMM_PER_INCH);

// Calculate the current height of the image in (0.01)mm
int pich = (int)Math.Round((_image.Height / yDpi) * HMM_PER_INCH);

...
```

Twips are screen-independent units used to ensure that the placement and proportion of screen elements in a screen application are the same on all display systems. The metafile's target dimensions in twips are calculated using the following conversion units and formula. (The example below explains how to find the target width, but the same formula is used to find the height by substituting height and vertical resolution for width and horizontal resolution respectively.)

1 Twip = 1/20 Point
1 Point = 1/72 Inch
1 Twip = 1/1440 Inch

[N]	= target width of the metafile in twips
	= Image Width in Inches * Number of twips per inch
	= (Image Width in Pixels / Graphics Context's Horizontal Resolution) * 1440
	= (Image Width in Pixels / Graphics.DpiX) * 1440

```
...

// Calculate the target width of the image in twips
int picwgoal = (int)Math.Round((_image.Width / xDpi) * TWIPS_PER_INCH);

// Calculate the target height of the image in twips
int pichgoal = (int)Math.Round((_image.Height / yDpi) * TWIPS_PER_INCH);

...
```

After the RTF representation of the image is created the image is inserted into the RTF document similarly to how text is inserted. If any text is selected, the image replaces the selected text. If no text is selected, the image is inserted at the location of the caret.

Using the code

To use the `ExRichTextBox` simply include the ExRichTextBox project as a reference in your project or compile the .dll and add it to your VS.NET Toolbox. There are two public properties that can be set: `TextColor` is the color that inserted text will have if no text color is specified when inserting; `HighlightColor` is the background color of inserted text if no highlight color is specified when inserting. By default, these properties are set to Black and White respectively. Two examples of using the control are included in the project download. The first simulates a chat window. The user can click an emoticon or type ":)" to insert a smiley (The sample only looks for the first occurrence of ":)"). It also illustrates how to insert plain text as RTF. The relevant methods are shown below.

⊟ **Collapse**

```
...

// When an emoticon is clicked, insert its image into to RTF
private void cmenu_Emoticons_Click(object _sender,
  EventArgs _args) {

  rtbox_SendMessage.InsertImage(_item.Image);

}

...

private void btn_SendMessage_Click(object sender,
  System.EventArgs e) {

  // Add fake message owner using insert
  rtbox_MessageHistory.AppendTextAsRtf("Local User Said\n\n",
    new Font(this.Font, FontStyle.Underline | FontStyle.Bold),
    RtfColor.Blue, RtfColor.Yellow);

  // Just to show it's possible, if the text contains a smiley face [:)]
  // insert the smiley image instead. This is not
  // a practical way to do this.
  int _index;
  if ((_index = rtbox_SendMessage.Find(":)")) > -1) {
    rtbox_SendMessage.Select(_index, ":)".Length);
    rtbox_SendMessage.InsertImage(
      new Bitmap(typeof(IMWindow), "Emoticons.Beer.png"));
  }

  // Add the message to the history
  rtbox_MessageHistory.AppendRtf(rtbox_SendMessage.Rtf);

  // Add a newline below the added line, just to add spacing
  rtbox_MessageHistory.AppendTextAsRtf("\n");

  // History gets the focus
  rtbox_MessageHistory.Focus();

  // Scroll to bottom so newly added text is seen.
  rtbox_MessageHistory.Select(rtbox_MessageHistory.TextLength, 0);
  rtbox_MessageHistory.ScrollToCaret();

  // Return focus to message text box
  rtbox_SendMessage.Focus();

  // Add the Rtf Codes to the RtfCode Window
  frm_RtfCodes.AppendText(rtbox_SendMessage.Rtf);

  // Clear the SendMessage box.
  rtbox_SendMessage.Text = String.Empty;
}
```

```
...
```

The second sample included is a way to check how the `ExRichTextBox` handles large images. A user can insert bitmaps, JPEGs, GIFs, Icons, PNGs, and TIFFs, or insert plain text, all from the menu.

Points of Interest

The `ExRichTextBox` is a good solution for inserting small images, but it takes a full 2.65 seconds to insert a 24bit, 432 X 567 JPEG into the second sample application. That's because the image is being copied to an array of bytes then to a string, then inserted. There should be a way to insert the byte representation of the image at a lower level, skipping the string conversion. However, the author is currently not that familiar with the Win32 API, so this will be an improvement in the near future.

Khendys Gordon

Click here to view Khendys Gordon's online profile.

Other popular C# Controls articles:

- Themed Windows XP style Explorer Bar
  A fully customizable Windows XP style Explorer Bar that supports Windows XP themes and animated expand/collapse with transparency.
- XPTable - .NET ListView meets Java's JTable
  A fully customisable ListView style control based on Java's JTable.
- SourceGrid - Open Source C# Grid Control
  SourceGrid is a free open source grid control. Supports virtual grid, custom cells and editors, advanced formatting options and many others features
- TaskbarNotifier, a skinnable MSN Messenger-like popup in C# and now in VB.NET too
  The TaskbarNotifier class allows to display an MSN Messenger-like animated popup with a skinned background

Dundas Chart for .NET
The INDUSTRY LEADER in .NET Charting Solutions
Ajax Enabled
NEW
Dundas Data Visualization

Msgs 1 to 25 of 106 (Total: 105) (Refresh) First Prev Next

Subject	Author	Date
In C++ : how to insert programmatically BMP or Tiff images while preserving the original size ?	**philbel**	**3:49 6 Sep '06**
work in vb.net	**ibless**	**19:50 1 Sep '06**
Bug: escape some special chars	**davidc912**	**23:41 9 Aug '06**
Links behind the icons	**fiechter**	**3:02 7 Jul '06**
Problems with inserting imge into RTF control	**Misiacik7**	**6:10 20 May '06**
Great work	**ChrisMatei**	**10:49 10 May '06**
new line issue	**TheMadMonkey**	**10:36 3 May '06**
Re: new line issue	TheMadMonkey	11:42 3 May '06
Open File Function?	**pkwan7**	**2:10 1 May '06**
Re: Open File Function?	pkwan7	2:18 1 May '06
Extract Image out of the RTF	**Esam Salah**	**13:04 7 Mar '06**
Re: Extract Image out of the RTF	Assaf Koren	2:35 22 Mar '06
Problem with url !!!	**Robin Mimeault**	**7:53 26 Jan '06**
Re: Problem with url !!!	Robin Mimeault	9:30 26 Jan '06
Great	**Faraz Masood**	**19:58 8 Jan '06**
Problem with images and backcolor	**Omega_Supreme**	**0:46 4 Dec '05**
great but?	**Squall3777**	**12:22 4 Nov '05**
Re: great but?	Robert G. Schaffrath	23:15 13 Jul '06
Please help with font issue	**AIU4840**	**19:23 17 Oct '05**
Re: Please help with font issue	AIU4840	11:16 18 Oct '05
Can't change the font	**shekky**	**22:54 16 Oct '05**
Re: Can't change the font	shekky	22:57 16 Oct '05
Re: Can't change the font	FaxedHead	8:58 15 Jan '06
Fix + Courier support	BoeroBoy	12:01 22 Feb '06
Contact	**Terry McGinty**	**15:56 10 Sep '05**

Last Visit: 15:17 Wednesday 13th September, 2006 First Prev Next

General comment message · News / Info · Question · Answer · Joke / Game · Admin

Updated: 15 Jul 2003