Developers Manual and API Reference

Table of Contents

1 Welcome to RTF2PDF / TextDynamic Server

Welcome to wRTF2PDF. This is not just a tool to convert RTF to PDF or HTML to PDF, it also includes a full featured, completely self contained word processing engine to create formatted text in code.

RTF2PDF V4 is available now. It is based on the word processing engine WPTools 7 and the PDF creation tool wPDF 4.

You can use this component to create documents and invoices on a web server and save the result as RTF, HTML or PDF file. Since it offers a very similar programming API as the word processing control TextDynamic it is also known as "TextDynamic Server".

The integrated word processing engine supports:
- Supports multiple character attributes (fonts, font sizes, colors, styles and colors)
- Paragraph attributes (indents, spacing, alignment, justified text)
- Tabstops (left, right, decimal, centered and fill signs)
- Paragraph styles with CSS support
- Tables, nested tables, row merge, column merge
- Footnotes
- Textboxes
- Columns
- Headers and Footers
- Images (BMP, JPEG, EMF, PNG), linked and embedded, also with wrap text on both sides!
- Hyperlinks and nestable bookmarks
- powerful mail merge

The integrated PDF engine supports
- Compression
- Encryption
- create PDF/A compliant files (I automatically adds meta tags to distinguish informative text from layout elements)
- embed fonts + subset fonts
- convert embedded EMF to PDF vectors
- With the "server" license You can export pages from multiple documents into one multipage TIFF file.

In case your development system does not support COM, You can use the rich text to PDF conversion also using some simple to use methods exported by the DLL.

Product Page RTF2PDF	Product Page PDFControl	API Reference	Support forum

Technical Note:

The component wRTF2PDF is based on our word processing component TextDynamic plus the PDF engine wPDFControl. It does everything what wPDFControl and much of what TextDynamic does. It has been optimized to work on a server, or webserver. It does not print but it can produce PDF and other file formats fast and effectively. It can also export documents page by page as metafile.
Please see our demo ASP web server http://178.77.68.45/.

We recommend to program with RTF2PDF similar to TextDynamic - but as an invisible, server control. Using the interface IWPEditor you have access to powerful and effective means to create text or tables in code.

While TextDynamic is a visual tool, RTF2PDF has been optimized to work in the background. It can be easily used with ASP or ASP.NET to create HTML or PDF response output. Since the API is so similar to the API of TextDynamic you can use almost the same code in your ASP page as you used for your desktop application. Due to its nature it does not offer any GUI elements and does not implement the wpa actions used by TextDynamic.

Instead of the interface IWPMemo which is implemented by TextDynamic; RTF2PDF implements the interface IWPEditor. It contains a subset of the IWPMemo methods and properties.

The RTF2PDF ActiveX interface can be used very similar to the .NET Interface.
We included some simple VB / VBS examples here...

Example to create the text "Hello World" in ASP.NET (language C#):

```
RTF2PDF pdf = New RTF2PDF();
// The main editor interface: load & save
WPDynamic.IWPEditor Memo = pdf.Memo;
// Modify the current writing attribute
WPDynamic.IWPAttrInterface CurrAttr = Memo.CurrAttr;
// TextCusror contains methods to insert paragraphs, tables, text, images, fields, header
WPDynamic.IWPTextCursor TextCursor = Memo.TextCursor;
// Clear the text
Memo.Clear();
// Set the current font
CurrAttr.SetFontface("Verdana");
CurrAttr.SetFontSize(10);
// Create some text
TextCursor.InputString("Hello World\r ",0);
```

Only the first two lines would be different when using TextDynamic. There you probably use a control dropped on the form (the default name is wpdllint1) and the interface name IWPMemo instead of IWPEditor!

The export to PDF as ASP Result can be done like this. No temporary file will be written!
(Requires "Internet Server" License)

```
IWPPDFCreator PdfCreator = pdf.PdfCreator;
// reformat the text when we have completely created it
Memo.ReformatAll(false,false);

// Set PDF properties. Note that the interface IWPPDFCreator is being used
PdfCreator.PDFFile = "memory"; // We want to use "ResultBuffer"
PdfCreator.FontMode = 0;  // i.e. embed all fonts
pdf.Print();

// remove any output
Response.Clear();
// Add new header
Response.ContentType = "application/pdf";
Response.AddHeader("Content-Type", "application/pdf");
// Set a file name which is displayed when download starts
Response.AddHeader("Content-Disposition","inline;filename=PortableDocument.pdf");
// Write the PDF data
Response.Clear();
Response.BinaryWrite(pdf.ResultBuffer);
```

2 Getting started with Rtf2Pdf / TextDynamic Server

wRTF2PDF / TextDynamic Server by WPCubed GmbH is a universal tool to convert RTF files into PDF. It supports embedded images, tables, headers and footers, sections, embedded images, even textboxes and foot notes.

The powerful API makes it easy to

- **load** text (RTF, ANSI, HTML, UNICODE and MIME)
- **process** text, create tables, insert text and do mail merge
- **generate** text (PDF, RTF, ANSI, HTML, UNICODE and MIME)

The object RTF2PDF is located in the assembly wPDF.dll. You need the RTF2PDF license to be able to use, the object will not work with wPDFControl.

When using RTF2PDF V3.5 or later we recommend to only use the property PDFCreator to change the way the PDF file is created. The "old" properties will still work though. Please also do not change the Command() function anymore to work with the text. Now you have a powerful API to create, load and save text. It is possible to load text in RTF, HTML, the proprietary WPT, ANSI and UNICODE format.

RTF2PDF can be easily used with ASP or ASP.NET to create HTML or PDF response output. Please check out our demo server. Since the API is so similar to the API of TextDynamic you can use almost the same code in your ASP page as you used for your desktop application. Due to its nature, it does not offer any GUI elements and does not implement the wpa actions used by TextDynamic.

RTF2PDF can also be used in VBS, VB and alike: Please see our introduction ActiveX example code!

Methods	Properties
BeginDoc - start a PDF document	**Memo** - provides an IWPEditor interface with many sub properties and -methods
EndDoc - closes a PDF document	**Memo2**
Print - exports the text in editor #1	**PDFCreator** - modify the PDF properties
PrintSecond - exports the text in editor #2. Use this to export a report!	**ResultBuffer** - read the created PDF data as array of bytes. PDFFile in PDFCreator must be set to "memory" Also see the TranferHelper class.
StartEngine /SetLicense - set license key and initialize.	**Report** - used for the optional reporting feature.
StopEngine - stops engine (OCX, C++ only)	AttrHelper - can be used to calculate a character attribute index to be used with TextCursor.InputString

Main interface for TextDynamic to manipulate text in editor #1 or #2:
IWPEditor/IWPMemo

Text creation and cursor positioning:
IWPTextCursor

Other Interfaces:

IWPAttrInterface	Update text attributes
IWPCharacterAttr	Attributes for links and fields
IWPDataBlock	Manage header and footer and text layers
IWPEditor	Main interface for RTF2PDF to manipulate text #1 or text #2
IWPFieldContents	Work with mail merge fields
IWPMapi	Interface to create and send e-mails
IWPMeasurePageParam	This interface is used by the OnMeasurePage event.
IWPNumberStyle	Interface to modify a numbering style.
IWPPageSize	Interface to update Pagesize
IWPPageSizeList	Use this list to create set of rectangles and fill them with text
IWPParInterface	Low level paragraph text and attribute access
IWPPdfCreator	Interface to start and configure PDF creation
IWPPicture	.NET Image2Picture utility class
IWPPrintParameter	Optimize printing (duplex, tray selection)
IWPStream	.NET Stream2WPStream utility class
IWPTextCursor	Text creation and cursor positioning
IWPTextObj	Change field, object and image properties
IWPTextWriter	Access writing properties

Not used are yet (reserved for optional reporting edition)

IWPReport	Manage reporting template
IWPReportBand	Manage report bands and groups
IWPReportVar	Manage reporting variables (to sum values)

Also see:
wPDFControl Methods and Properties

2.1 Overview

Note: Instead of the interface IWPMemo which is implemented by TextDynamic, RTF2PDF implements the interface IWPEditor. It contains a subset of the IWPMemo methods and properties. Please note that RTF2PDF does not print text.

The interfaces (IWPEditor, IWPTextCursor, ...) are the same in the .NET assembly and in the OCX. This makes it possible to make the source code exchangeable. In this reference we mainly include examples developed in C#.

Usually you will access the interfaces like this:

```
IWPMemo Memo = wpdllint1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;

TextCursor.InputText("Hello World");
Memo.Reformat();
```

Unless you use the demo, you will need to use

public bool SetLicense(string Name, string Code, uint Number);

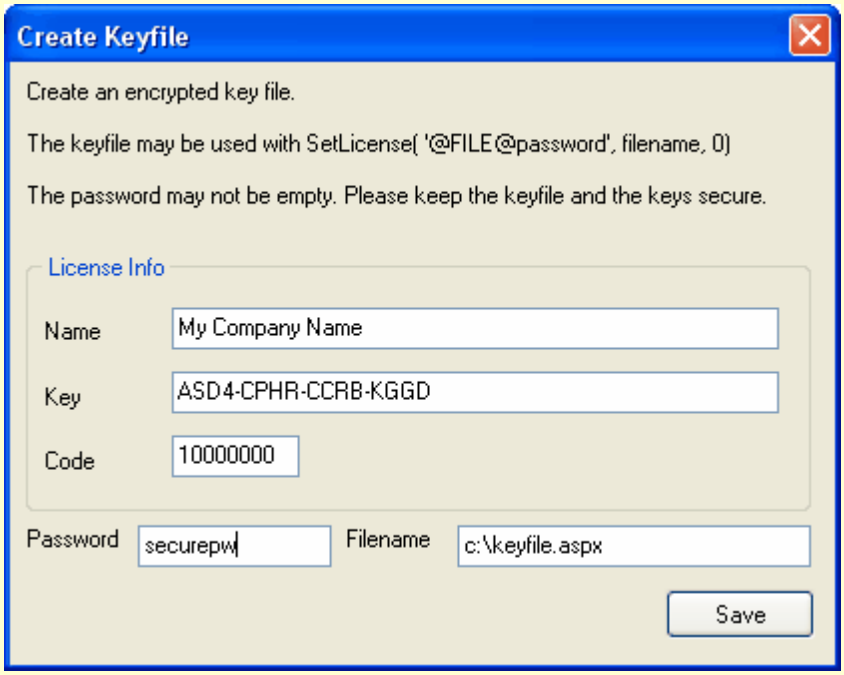
to tell the engine your license data.

You need to use a key which looks like wwww-xxxx-yyyy-zzzz.
The old code/name combination will not work for RTF2PDF 3.x "Plus"

When using ASP it is useful to load the license from a file

SetLicense("@FILE@securepw", "c:\\keyfile.aspx", 0)

The key file can be created with the demo application: Create Keyfile



Please make sure this file and password is secured.
Please make sure the IIS_IUSRS have read access to the license file, otherwise the following code will fail.

2.2 QuickStart

The conversion of a RTF file to PDF does not require much code.

(Please note that we have created hyperlinks for the important types to their description.)

A) Convert just a RTF, HTML or MSG file into PDF - C# Code

```
rtF2PDF1.SetLicense("","",0); // Your data here ....
rtF2PDF1.FileName = "c:\\newpdf.pdf";
if(!rtF2PDF1.LoadRTF( TemplateName.Text ))
   MessageBox.Show("Cannot load " + TemplateName.Text, "");
else
{
```

```
    rtF2PDF1.Export();
}
```

B) Use the new TextDynamic Interfaces

a) Export a File (VS2008 Demo) (use IWPMemo)

```
using wPDF;
using WPDynamic;

namespace TestReport
{
    class Program
    {
        static void Main(string[] args)
        {
            RTF2PDF pdf = null;
            try
            {
                pdf = new RTF2PDF();
                pdf.SetLicense("...", "...", xxx);
                pdf.Memo.LoadFromFile("c:\\Description.RTF");
                pdf.Export("c:\\temp\\Description.pdf");
            }
            finally
            {
                pdf.Dispose();
            }
        }
    }
}
```

b) Convert HTML or mime encoded emails (HTM, MSG, MHT, EML)

```
using wPDF;
using WPDynamic;

namespace TestReport
{
    class Program
    {
        static void Main(string[] args)
        {
            RTF2PDF pdf = null;
            try
            {
                pdf = new RTF2PDF();
                pdf.SetLicense("...", "...", xxx);

                // Activate HTML specific formatting
                pdf.Memo.SetBProp(BPropSel.wpViewOptions, 0, 1);
                // Load a Mime encode e-mail
                pdf.Memo.LoadFromFile("c:\\Test.EML");
                pdf.Export("c:\\temp\\Mail.pdf");
            }
            finally
            {
```

```
                pdf.Dispose();
            }
        }
    }
}
```

c) Create some text "on the fly" (use IWPTextCursor)

```
RtF2PDF1.SetLicense("...","xxxx-yyyy-zzzz-0000",12345);
RtF2PDF1.Memo.LoadFromFile(@"c:\testfile.rtf",false,"AUTO");

// requires: using wPDF; using WPDynamic; - we add some text
IWPTextCursor Cursor;
Cursor = RtF2PDF1.Memo.TextCursor;
Cursor.CPPosition = 0;
Cursor.InputString("Some text at the start\r", 0);

// Now create PDF file
RtF2PDF1.PdfCreator.PDFFile = @"c:\temp\\newfile.pdf";
RtF2PDF1.PdfCreator.Print();
```

d) Create Bookmarks

PDF Bookmarks are the optional table of content entries. RTF2PDF is able to create those strings automatically from the loaded text. To do so the paragraphs which should appear in this list should be marked with the property WPAT.ParIsOutline.

You can use the Memo.TextCommandStr to mark paragraphs which either use a certain style or which start with certain text to be used in the outline.

To create the TOC as text (within the document) use

```
Cursor.CPPosition = 0xFFFFFF;
Cursor.InputParagraph(0,"");
rtF2PDF1.Memo.TextCommand(2, 2);
```

C) Simple ASP.NET demo which sends the PDF as response

```
RTF2PDF pdf = new RTF2PDF();
pdf.Memo.LoadFromFile("somtext.rtf", true, "AUTO");
pdf.PdfCreator.PDFFile = "memory";
// remove any output
Response.Clear();
// Add new header
Response.ContentType = "application/pdf";
Response.AddHeader("Content-Type", "application/pdf");
// Set a file name which is displayed
Response.AddHeader("Content-Disposition","inline;filename=PortableDocument.pdf");
// Write the PDF data
Response.Clear();
// Finish
pdf.Dispose();
```

D) Activate Format mode optimized for HTML

a) Use Memo.SetBProp
Group 7 : wpAsWebPage

```
b) Use RtF2PDF1.Command(1312, 1) // WPCOM_SELECT_HTML_MODE = 1312;
```

E) Use the optional reporting feature

```
using wPDF;
using WPDynamic;

private void button1_Click(object sender, System.EventArgs e)
{
   // Set License with Reportion Option
   rtF2PDF2.SetLicense( "---", "---", 0);

   // Load a File
   rtF2PDF2.Memo.LoadFromFile(@"c:\Template_With_Tokens.RTF", false, "");

   // Use the Token To Template Conversion
   rtF2PDF2.Memo.TextCommandStr(17,0, "");

   // Create the Report (will be in Memo2)
   rtF2PDF2.Report.CreateReport();

   // Set Output Filename
   rtF2PDF2.FileName =  @"c:\test.pdf";

   // And convert Memo2 to PDF
   rtF2PDF2.PrintSecond();
}
```

We are also using this event to loop each group 10 times:

```
private void rtF2PDF2_OnReportState(object Sender, string Name, int State, WPDynamic.IWPR
{
      if (State==0)
      Abort = Band.Count>10;
}
```

F) VB6 Example using BeginDoc/EndDoc

```
  Dim Memo As IWPEditor ' TextDynamic uses IWPMemo.
  PDFControl1.StartEngine "S:\S\wPDFControl\wPDFControlDemo.dll", "lic", "lic", 0
  Set Memo = PDFControl1.Memo
  Memo.LoadFromFile "sometext.rtf", False, "AUTO"
  ' Create PDF File
  PDFControl1.BeginDoc "new2.pdf", 0
  ' We cannot call "Print" in VB6
  PDFControl1.ExecIntCommand 1305, 0
  PDFControl1.EndDoc
```

G) VB6 Example: wRTF2PDF using PDFCreator:

```
  Dim Memo As IWPEditor ' TextDynamic uses IWPMemo.
  Dim PDFCreator As IWPPdfCreator
  ' TODO: Modify license codes
  PDFControl1.StartEngine "wPDFControlDemo.dll", "lic", "lic", 0
  Set Memo = PDFControl1.Memo
  Set PDFCreator = PDFControl1.PDFCreator
  Memo.LoadFromFile "sometext.rtf", False, "AUTO"
  PDFCreator.PDFFile = "new.PDF"
```

```
  ' PDFCreator.Print - this cannot be used due to VB "bug"
  ' Use command instead
  PDFControl1.ExecIntCommand 1313, 0
```

2.3 Troubleshooting .NET

1) References

```
using wPDF;
using WPDynamic;
```

also add a reference to the correct wPDF3.DLL.
(There is one for the demo and one for the full version)

2) Copy Engine to the bin folder

For the demo wPDFControlDemo.DLL, for the full version wRTF2PDF03.DLL

3) With full version make sure to set the license key. The License key must enable RTF (and optional Reporting and/or Server) features

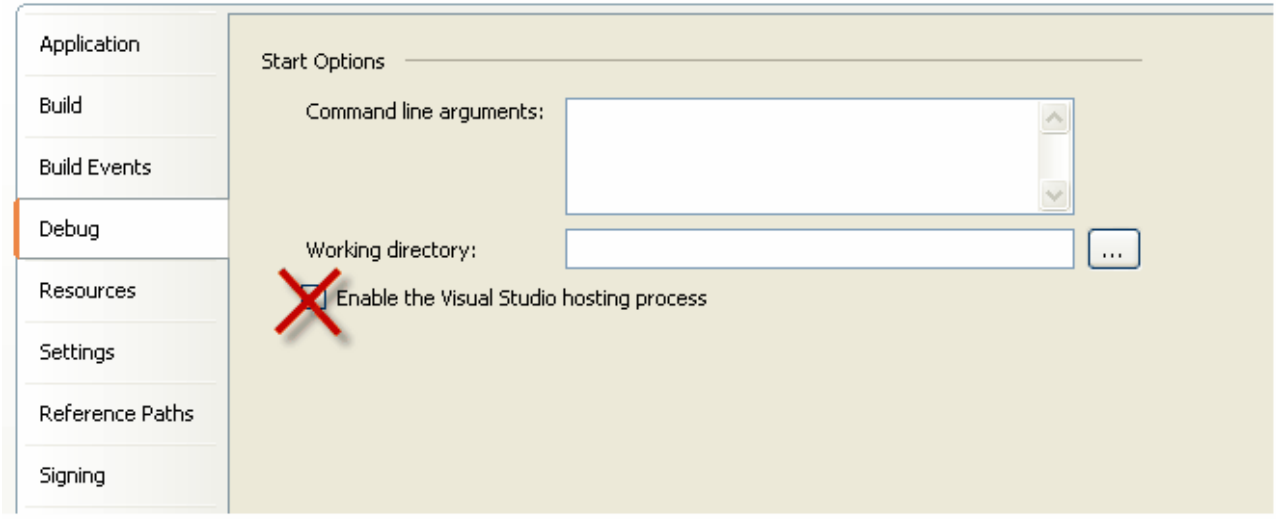
```
pdf.SetLicense("...", "...", xxx );
```

4) If you need to run it on a server and need the best possible threading performance

Please use special RTF2PDF Server License. It is at least 25% faster due to optimized code.

Troubleshooting

In case You use the wRTF2PDF demo DLL and see the error "Runtime 207" when closing the application please deactivate the Visual Studio 2008 hosting process by removing the check here:



Error: Cannot load DLL ...
Please open the project options and, under "Build", "Destination Platform" select "X86" instead of "X64" or "Any CPU".

Note: RTF2PDF V4 and TextDynamic V7 also include 64 bit engine DLLs.

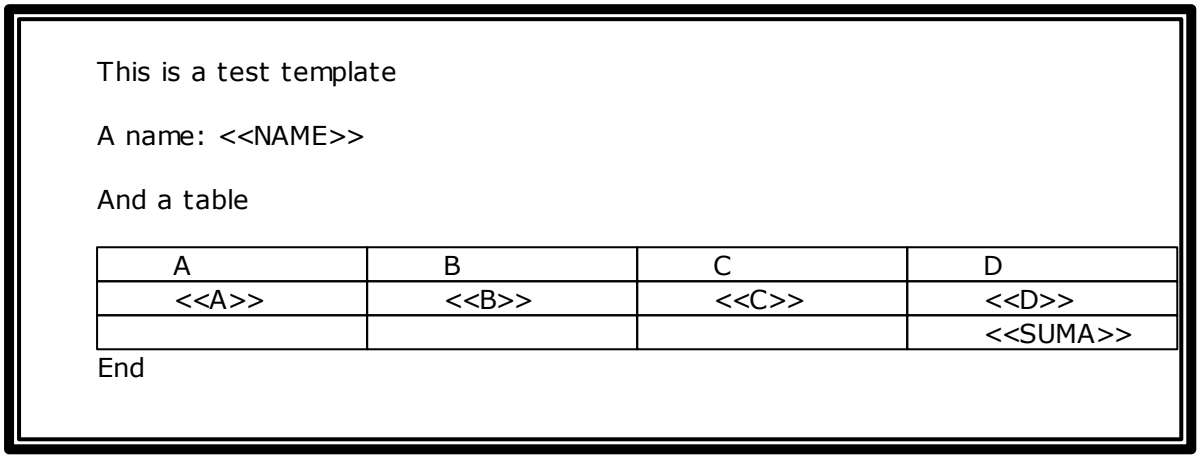
If you uncertain which DLL is beeing used, You can use this code:

```
Memo.TextCommand(30,0)
```

This will add the name of the module, its version and if it is 32 or 64 bit to the text.

2.4 Load Template, Update and Save

Here we work with a template which does not only contain data field placeholder but also tables.



This is a test template

A name: <<NAME>>

And a table

A	B	C	D
<<A>>	<<B>>	<<C>>	<<D>>
			<<SUMA>>

End

We can export this template using a few lines of code:

```
rtF2PDF1.SetLicense("","",0); // Your data here ....
rtF2PDF1.FileName = "c:\\newpdf.pdf";
if(rtF2PDF1.LoadRTF( "c:\Template.rtf" ))
{
   rtF2PDF1.Export();
}
```

To also update the fields we first convert the tags into merge fields and then step through the document and modify it accordingly.

```
// requires:  "using wPDF; using WPDynamic;"
IWPEditor Memo =  rtF2PDF1.Memo;
IWPTextCursor Cursor = Memo.TextCursor;

// Convert all <<*>> int merge fields since such fields can be updated a lot easier
// then deletion and inserting text. Also text attributes are preserved better.
Cursor.FieldsFromTokens("<<", ">>", "");

// Now we go through the text and update it

// Go to start
Cursor.CPPosition = 0;

// And move from mail merge field to mail merge field
while(Cursor.CPMoveNextObject(WPDynamic.TextObjTypes.wpobjMergeField, false))
{
   Memo.CurrObj.EmbeddedText = "data of " + Memo.CurrObj.Name;
}
```

Since this code now works to update the fields we update the code to modify each table to have as many data rows as records in a query. In this example however we just set the rowcount value to a random value. This code should be executed after FieldsFromTokens.

```
// Now we locate all tables and update their row count to match our query.
// In this example we just use a random value
Random rnd = new Random();
Cursor.CPPosition = 0;
while(Cursor.CPMoveNextTable())
{
   int rowcount = (int)rnd.Next(20);
   if (rowcount==0) Cursor.TableDelete();
   else
   {
      Cursor.CPTableRowNr = 1; // goto row # 1
      Memo.CurrPar.SetPtr( Cursor.CPRowPtr );
      while(rowcount-->0) Memo.CurrPar.Duplicate();
   }
}
```

2.5 ASP.NET Example

File Webform1.aspx.cs

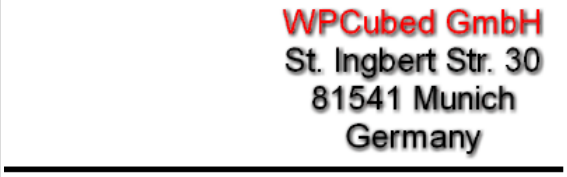
```
The "using" sequence
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
link assembly "wPDF" (from assembly wPDF3.DLL - please add to references)
using wPDF;
using WPDynamic;

namespace RTF2
{
The event is executed by ASP
public class WebForm1 : System.Web.UI.Page
      {
            private void Page_Load(object sender, System.EventArgs e)
            {

We create a RTF2PDF instance
                RTF2PDF pdf = new RTF2PDF();
                string password = "SomePrivatePassword"
                try
                {
                  // Load the license from file which is not in the www path
```

```
                // Please make sure the IIS_IUSRS have read access to this
file, otherwise the following code will fail.
                pdf.SetLicense("@FILE@" + password,"C:\\License\\rtflicense.
dat",0);
```

clear all properties and set the default font

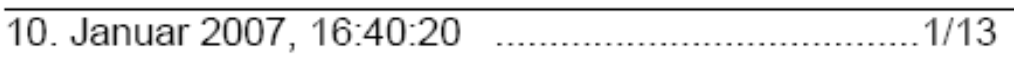
```
                pdf.Memo.TextCursor.Clear();
                pdf.Memo.CurrAttr.SetFontface("Verdana");
```

create the header text



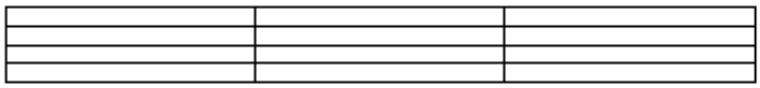

```
                pdf.Memo.TextCursor.InputHeader(0,"","");
                pdf.Memo.CurrAttr.SetFontSize(8);
                pdf.Memo.TextCursor.InputImage(
                       "c:\\inetpub\\wwwroot\\adr.png",0);
                pdf.Memo.CurrObj.ScaleSize(0,567*2,0); // Scale to height =
2 cm
                pdf.Memo.CurrPar.Alignment = 2; // right
                pdf.Memo.CurrPar.SpaceAfter = 30; // Distance between image
and line
                pdf.Memo.TextCursor.InputString("\r",0); // one line
                pdf.Memo.CurrPar.Borders = 2; // Top Border
                pdf.Memo.CurrPar.ParASet(
                       (int)WPDynamic.WPAT.BorderWidth, 30 ); // 1pt border
width
```

create the footer text




```
                pdf.Memo.TextCursor.InputFooter(0,"","");
                pdf.Memo.CurrAttr.SetFontSize(8);
                // Current Date
                pdf.Memo.TextCursor.InputFieldObject(
                       "DATE","\\@ \"dd. mmmm yyyy\"","");
                pdf.Memo.TextCursor.InputString(", ",0);
                pdf.Memo.TextCursor.InputFieldObject("TIME","","");
                // right tab in margin, dot filling
                pdf.Memo.TextCursor.InputTabstop(true,0xFFFF,1,1);

                pdf.Memo.TextCursor.InputString("\t",0);
                pdf.Memo.TextCursor.InputFieldObject("PAGE","","");
                pdf.Memo.TextCursor.InputString("/",0);
                pdf.Memo.TextCursor.InputFieldObject("NUMPAGES","","");
                pdf.Memo.CurrPar.Borders = 2; // Top Border
```

create some text, in this case just a table.



```
                // Fill the body
                pdf.Memo.TextCursor.GotoBody();
                pdf.Memo.CurrAttr.SetFontSize(10);
                pdf.Memo.TextCursor.InputString(
                        "\rThis Table was created in ASP.NET ",0);
```

```
                    pdf.Memo.CurrAttr.IncludeStyles(1); // <<bold

                    pdf.Memo.TextCursor.InputParagraph(0,"");
                    pdf.Memo.TextCursor.InputParagraph(0,"");
                    pdf.Memo.CurrAttr.ExcludeStyles(1); // >>bold

                    pdf.Memo.TextCursor.AddTable("",3,4,true,0,false, false);
```

Add text after the table.

```
                    // Create a new paragraph AFTER the table (mode=2)
                    pdf.Memo.TextCursor.InputParagraph(2,"");
                    pdf.Memo.TextCursor.InputParagraph(2,"");

                    pdf.Memo.TextCursor.InputString(
                           "Click to see the C# source: ",0);
```

Create an icon. When user clicks on it the C# source is opened!

Click to see the C# source: 


```
                    pdf.Memo.TextCursor.InputImage(
                         "c:\\inetpub\\wwwroot\\cs_logo.png",0);
                    pdf.Memo.CurrObj._SetObjType(100);
                    pdf.Memo.CurrObj.ScaleSize(567,0,0); // Scale to width = 1
cm
                    pdf.Memo.CurrObj.LoadFromFile(
                         "c:\\inetpub\\wwwroot\\WebForm1.aspx.cs");
```

Add some paragraphs
store current attributes

```
                      pdf.Memo.TextCursor.InputParagraph(2,"");
                    pdf.Memo.TextCursor.InputParagraph(2,"");
                    int save_ca = pdf.Memo.CurrAttr.CharAttrIndex;
```

add some text and modify the writing mode

```
                    pdf.Memo.TextCursor.InputString(
                         "This is the RTF data for the table:\r",0);
                    pdf.Memo.CurrAttr.SetFontSize(8);
                    pdf.Memo.CurrAttr.SetFontface("Courier New");
```

Now save the current text as RTF and insert it here as ANSI text.

This is the RTF data for the table:
{\rtf1\ansi\deff0\ucl\ansicpg1252\d

```
                    pdf.Memo.LoadFromString(
                       pdf.Memo.SaveToString(false, "RTF-nobinary"),
                          true, "ANSI"
                    );
```

Restore current writing mode

```
                    // We restore the character attributes used before
                    pdf.Memo.CurrAttr.CharAttrIndex = save_ca;
```

and add an icon to open this document in RTF format.

```
                    // And create an icon to attach the RTF source
                    pdf.Memo.TextCursor.InputString("\r\rClick to open the
document: ",0);
                    pdf.Memo.TextCursor.InputImage("c:\\inetpub\\wwwroot\
\rtf_logo.png",0);
                    pdf.Memo.CurrObj._SetObjType(100);
                    pdf.Memo.CurrObj.ScaleSize(567,0,0); // Scale to width = 1
cm
```

We create a stream,

```
save the document to this stream and load it into the "RTF" icon.
                    System.IO.Stream str = new System.IO.MemoryStream();
                    pdf.Memo.SaveToStream(
                         new WPDynamic.Stream2WPStream(str) ,false,"RTF");
                    pdf.Memo.CurrObj.LoadFromStream("Document.RTF",
                         new WPDynamic.Stream2WPStream(str));
Select the "in memory" PDF creation mode on web server (requires "Internet Server"
License)
                    pdf.PdfCreator.PDFFile = "memory";
                    pdf.PdfCreator.FontMode = 0;  // embed all fonts!
                    try
                    {
Reformat the text
                    pdf.Memo.ReformatAll(false,false);
And create PDF
                    pdf.Print();
Initialize the HTTP headers.
                    // remove any output
                    Response.Clear();
                    // Add new header
                    Response.ContentType = "application/pdf";
                    Response.AddHeader("Content-Type", "application/pdf");
                    // Set a file name which is displayed
                    Response.AddHeader("Content-Disposition","inline;
filename=PortableDocument.pdf");
                    // Write the PDF data
                    Response.Clear();
and use "ResultBuffer" to assign the Response
                      Response.BinaryWrite(pdf.ResultBuffer);
                }
            catch
                {
                  Response.Write("Internal Error in PDF Creation");
                }
            }
            finally
            {
                  pdf.Dispose(); // Don't forget!
                  Response.End();
            }

            }

            #region automatically created code [...]
      }
}
```

File test.aspx contains a single line:

```
<%@ Page Language="c#" Debug="true" Codebehind="/WebForm1.aspx.cs" AutoEventWireup="false
```

To work with ASP.NET please copy the engine DLL (RT2PDF03.DLL) and the DLL wPDF.DLL into the directory bin.

Note: With wRTF2PDF V4 You will need the DLLs wPDF4.DLL and wRT2PDF04W.DLL.

In our demo the \bin directory contains:

The file in the root directory (wwwroot) are

Note: Often only the compiled DLL (here RTF2.DLL) is required for the ASP program. The other files which may be created by your developing tool are not created and can confuse the web server. The one line test.aspx mentioned above was enough to start this program!

2.6 ASP.NET Live Demos

On our test server server at **www.rtf-net.com** (or http://178.77.68.45/rtf/) we host some asp.net demos.

You can click on the button 

and the underlying C# source will be displayed as PDF page. Naturally this PDF was created from the current source code which was formatted and exported by RTF2PDF! Using the other buttons you can see the **output** of the demo, in PDF optionally also in HTM and RTF format.

RTF2PDF/TextDynamic Server by WPCubed GmbH

This page is dedicated to show life ASP.NET examples, it runs on a virtual server (Windows Server 2008, 256 MB RAM, 3 GB HD).

RTF2PDF/TextDynamic Server is the ultimate tool to create documents (RTF, HTML, PDF, MIME) using ASP or ASP.NET. The API now provided by RTF2PDF is almost the same as the API of our visual word processing component TextDynamic . This makes it possible to use similar code for either product. RTF2PDF includes .NET assemblies for .NET 3.5 and .NET 4 and an ActiveX. It can still be used in WinForms and Visual Basic 6. The basic RTF to PDF conversion functionality can be accessed directly in the DLL.

1) The demo **"showcs"**formats the source of our example projects. Of course it uses RTF2PDF itself. It loads the original source code into RTF2PDF, removes the redundant parts and formats it to make it easier to read. It then adds some additional text plus header and footer.
The resulting document is passed back as PDF data when View Source was clicked.
(This project is always used when you click on "View Source" for any of the other projects.)

2) The demo **"crtable"**shows how to create a text in code. It first creates a table using the AddTable API and fills in some random numbers later. The result can be published as RTF, PDF and HTML code.

2) The demo **"images"**shows how to create an image in the text. Since we are also creating an embedded metafile "on the fly" (a screenshot of the document itself (created "on the fly"!) there is no HTML output for this demo. The RTF was exported with embedded images.

2.7 Properties

2.7.1 Report

Applies to
Rtf2Pdf

Declaration
```
IWPReport Report;
```

Description
Reporting is an optional feature which is activated by the license key.

Usually you will only use the "Token to Template Conversion".

2.7.2 Memo

Applies to
Rtf2Pdf

Declaration
```
IWPEditor Memo;
```

Description
This is the interface to load text into the editor #1. With RTF2PDF it uses the interface IWPEditor while TextDynamic uses IWPMemo. IWPEditor implements a subset of the methods in IWPMemo since some do not make sense for an invisible control. You can use Memo.TextCursor

to add text using code.

If access to Memo fails, probably the license was not set. Please see SetLicense.

2.7.3 Memo2

Applies to
Rtf2Pdf

Declaration
IWPEditor `Memo2;`

Description
This is the interface to load text into the editor #2.

With RTF2PDF it uses the interface IWPEditor while TextDynamic uses IWPMemo. IWPEditor implements a subset of the methods in IWPMemo since some do not make sense for an invisible control. You can use Memo.TextCursor to add text using code.

2.7.4 PDFCreator

Applies to
Rtf2Pdf

Declaration
`IWPPdfCreator PDFCreator;`

Description
This interface can be used to change the properties of the PDF engine. We recommend to use this interface instead of the properties provided on class level since this makes the code compatible to TextDynamic.

2.7.5 ResultBuffer

Applies to
Rtf2Pdf

Declaration
```
property ResultBuffer: Variant;

byte[] ResultBuffer;
```

Description
This property provides the result of the PDF creation as memory buffer (variant array of bytes).

Also see the "ASP TransferHelper" - it avoids flooding the server.

Note: RTF2PDF may be only used on a web server when you have the **Server License** of RTF2PDF / wPDFControl!

Example: (Response is the ASP object defined in event Page_Load.)

```
// Initialize pdf creation
rtf2pdf.PdfCreator.PDFFile = "memory";
rtf2pdf.PdfCreator.FontMode = 0;  // embed all fonts!
rtf2pdf.Memo.ReformatAll(false,false);
rtf2pdf.Print();
// remove any output
Response.Clear();
// Add new HTTP header
Response.ContentType = "application/pdf";
Response.AddHeader("Content-Type", "application/pdf");
// Set a file name which is displayed
Response.AddHeader("Content-Disposition","inline;filename=PortableDocument.pdf");
// Write the PDF data
Response.Clear();
Response.BinaryWrite(rtf2pdf.ResultBuffer);
```

2.7.6 AttrHelper

Applies to
Rtf2Pdf

Declaration
`IWPAttrInterface AttrHelper;`

Description
This attribute interface can be used to interpret and create character attribute index values. This code first assigns the attribute index values which represents the character attributes "Courier New, 10.5pt, green" to an integer variable and then inserts text which will use this attributes

```
IWPAttrInterface help = wpdf.AttrHelper;
help.Clear();
help.SetFontface("Courier New");
help.SetColor(wpdllInt1.ToRGB(Color.Green));
help.SetFontSize(10.5F);
int standard = help.CharAttrIndex;
wpdllInt1.TextCursor.InputString("Hello World", standard);
```

Category
Character Attributes

2.8 Methods

2.8.1 BeginDoc

Applies to
Rtf2Pdf

Declaration
`function BeginDoc(const FileName: WideString; UseStream: Integer): Integer;`

Description
This method starts a PDF document.

2.8.2 EndDoc

Applies to
Rtf2Pdf

Declaration
```
procedure EndDoc;
```

Description
The method closes a PDF document started with BeginDoc.

2.8.3 Print

Applies to
Rtf2Pdf

Declaration
```
function Print: Integer;
```

Description
Use this method to print the contents of editor #1. This is the the same as using PDFCreator. Print();

VB6 Note:

If you are using VB6 you cannot use this method since Visual basic overrides this name with special functionality.

You need to use RTF2PDF.ExecIntCommand 1305, 0 instead.

Example:

```
Dim Memo As IWPEditor ' TextDynamic uses IWPMemo.
PDFControl1.StartEngine "S:\S\wPDFControl\wPDFControlDemo.dll", "lic", "lic", 0
Set Memo = PDFControl1.Memo
Memo.LoadFromFile "sometext.rtf", False, "AUTO"
' Create PDF File
PDFControl1.BeginDoc "new2.pdf", 0
' We cannot call "Print" in VB6
PDFControl1.ExecIntCommand 1305, 0
PDFControl1.EndDoc
```

2.8.4 PrintSecond

Applies to
Rtf2Pdf

Declaration
```
function PrintSecond: Integer;
```

Description
Use this method to print the contents of editor #2. This is the the same as using PDFCreator. PrintSecond();

2.8.5 StartEngine / SetLicense

Applies to
Rtf2Pdf

used by OCX:
```
function StartEngine(const DLLName: WideString; const LicenseName: WideString;
const LicenseCode: WideString; LicenseNumber: Integer): WordBool;
```

used by .NET Assembly:
public bool SetLicense(String Name, String Code, uint Number)

This method must be used to start the PDF and text creation.

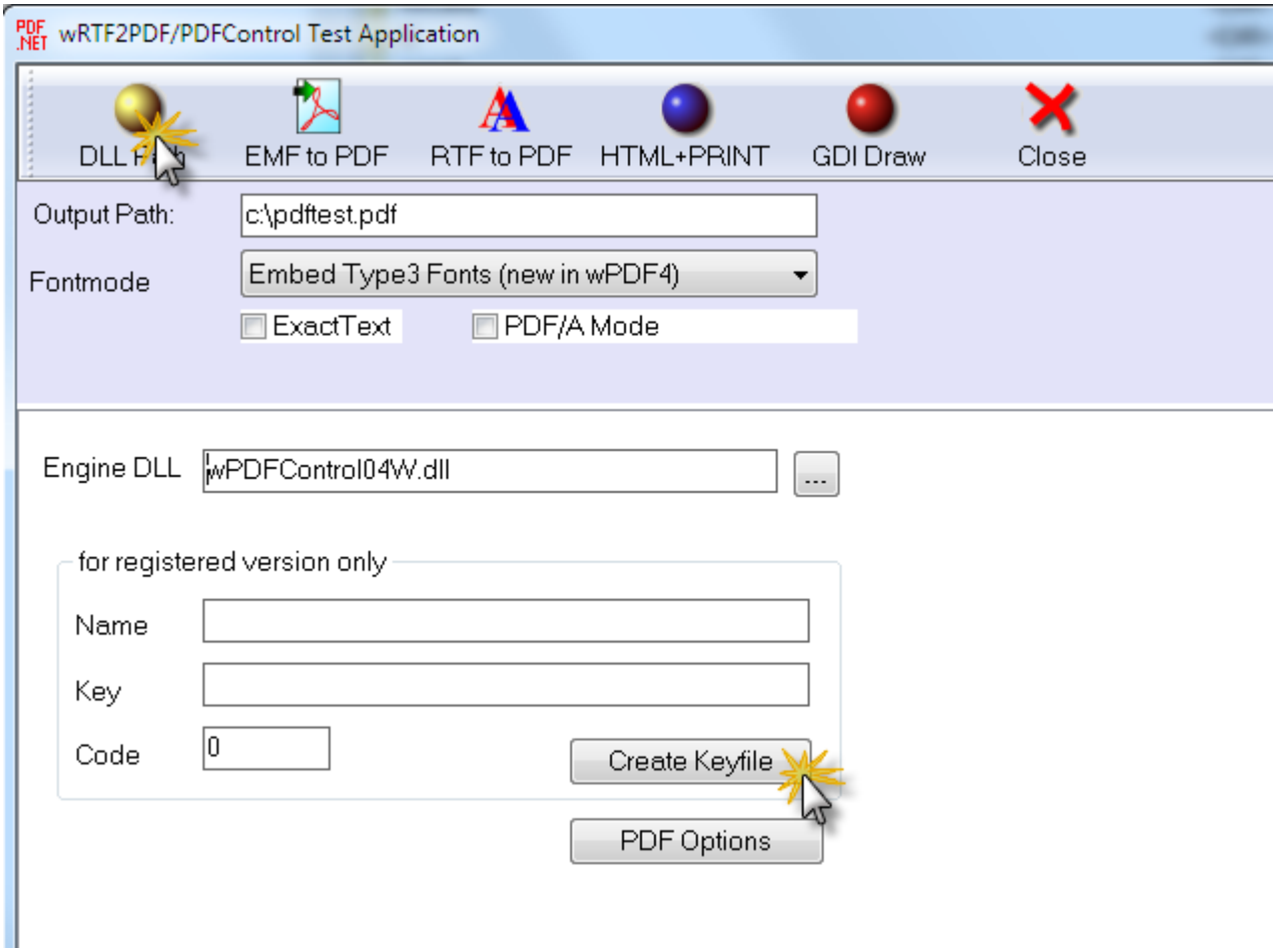
It is also possible, and recommended for ASP use, to load the license info from a file.
In this case LicenseName must be @FILE@*password*, and LicenseCode the filename. The password can be defined when created the key file.

Example:

```
pdf.SetLicense("@FILE@" + password,"C:\\License\\rtflicense.dat",0);
```

Please make sure the IIS_IUSRS have read access to this file, otherwise the following code will fail.

The license file can be created inside the provided demo exe:



2.8.6 StopEngine

Applies to
Rtf2Pdf

Declaration
```
procedure StopEngine;
```

Description
This method de allocates the memory which was initialized by StartEngine.

Only used by OCX and native C interface.

2.8.7 ReleaseInt

This method (it is exclusive to the .NET assembly) releases a reference to a programming interface which was retrieved from TextDynamic.

It is necessary to use this method at the end of the methods which use an interface to make sure the garbage collection frees the interface at once. Otherwise, when the editor has been disposed already, it can happen that the garbage collection accesses memory which has been freed.

Example:
```
IWPMemo Memo = wpdllint1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;

TextCursor.InputText("Hello World");
Memo.Reformat();

wpdllint1.ReleaseInt(TextCursor);
wpdllint1.ReleaseInt(Memo);
```

Note: All interfaces passed as event parameters are automatically released.

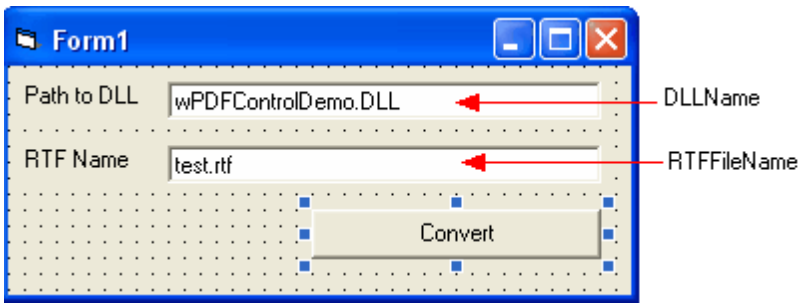
2.9 RTF2PDF ActiveX

The RTF2PDF ActiveX interface can be used very similar to the .NET Interface.

Note: In VB6 you will have to execute **command #1305** to export the text, the method "Print" is overridden by VB6.

Here we included some simple VB / VBS examples:

In Visual Basic 6 we created this form:



For to launch the example code.

To make it compatible to VBS we create the ActiveX at runtime.

From the button we call the following code. It uses the old style "commands" which were used until RTF2PDF V3.5 was released.

```
Private Sub Command1_Click()
  ' Initiate the OCX
  Set PDF = CreateObject("wPDF_X01.PDFControl")
  ' Start Engine, last param =LIC_CODE
  If PDF.StartEngine(DLLNAME.Text, "LIC_NAME", "LIC_KEY", 0) Then
  ' Start a PDF document
  If PDF.BeginDoc(Replace(RTFFileName.Text, ".rtf", ".pdf"), 0) Then
    ' Before RTF2PDF V3.5 you would have used this commands:
    PDF.ExecIntCommand 1000, 0 ' init RTF conversion
    PDF.ExecStrCommand 1002, RTFFileName.Text ' load RTF file
    PDF.ExecIntCommand 1100, 0 ' convert it
    ' Finish PDF creation
    PDF.EndDoc
    PDF.StopEngine
  Else
    MsgBox "Cannot start PDF file"
  End If
  Else
    MsgBox "Cannot load RTF2PDF Engine from " + DLLNAME.Text
  End If
End Sub
```

With RTF2PDF V3 we can use the integrated interfaces which make it easy to not only convert a file but also change it before the conversion.

```
Private Sub Command1_Click()
  ' Initiate the OCX (if it is not already on the form)
  Set PDF = CreateObject('wPDF_X01.PDFControl')
  ' Start Engine, last param =LIC_CODE
  If PDF.StartEngine(DLLNAME.Text, "LIC_NAME", "LIC_KEY", 0) Then
    ' Load the file, TRUE would insert/append it
    If PDF.Memo.LoadFromFile(RTFFileName.Text, False, "AUTO") Then
      PDF.PDFCreator.PDFFile = Replace(RTFFileName.Text, ".rtf", ".pdf")
      ' Convert it
      ' VB6 does not let us call PDF.PDFCretor.Print so we use a command instead
      PDF.ExecIntCommand 1305, 0
      ' Finish PDF creation
    Else
       MsgBox "Cannot load RTF file " + RTFFileName.Text
    End If
    PDF.StopEngine
  Else
    MsgBox "Cannot load RTF2PDF Engine from " + DLLNAME.Text
  End If
End Sub
```

We added some code to replace some text and select landscape orientation

```
    If PDF.Memo.LoadFromFile(RTFFileName.Text, False, "AUTO") Then
      ' replace text
      PDF.Memo.TextCursor.ReplaceText "[NAME]", "Julian Ziersch", True, False, False, Tru
      ' Chage Pagesize
```

```
  PDF.Memo.PageSize.Landscape = True
.....
```

2.10 wPDFControl Documentation

The component wPDFControl was created to make it easy for programmers to add PDF support to their application. It is based on the popular PDF engine 'wPDF' which has been successful on the Delphi and C++Builder market since the year 2000.

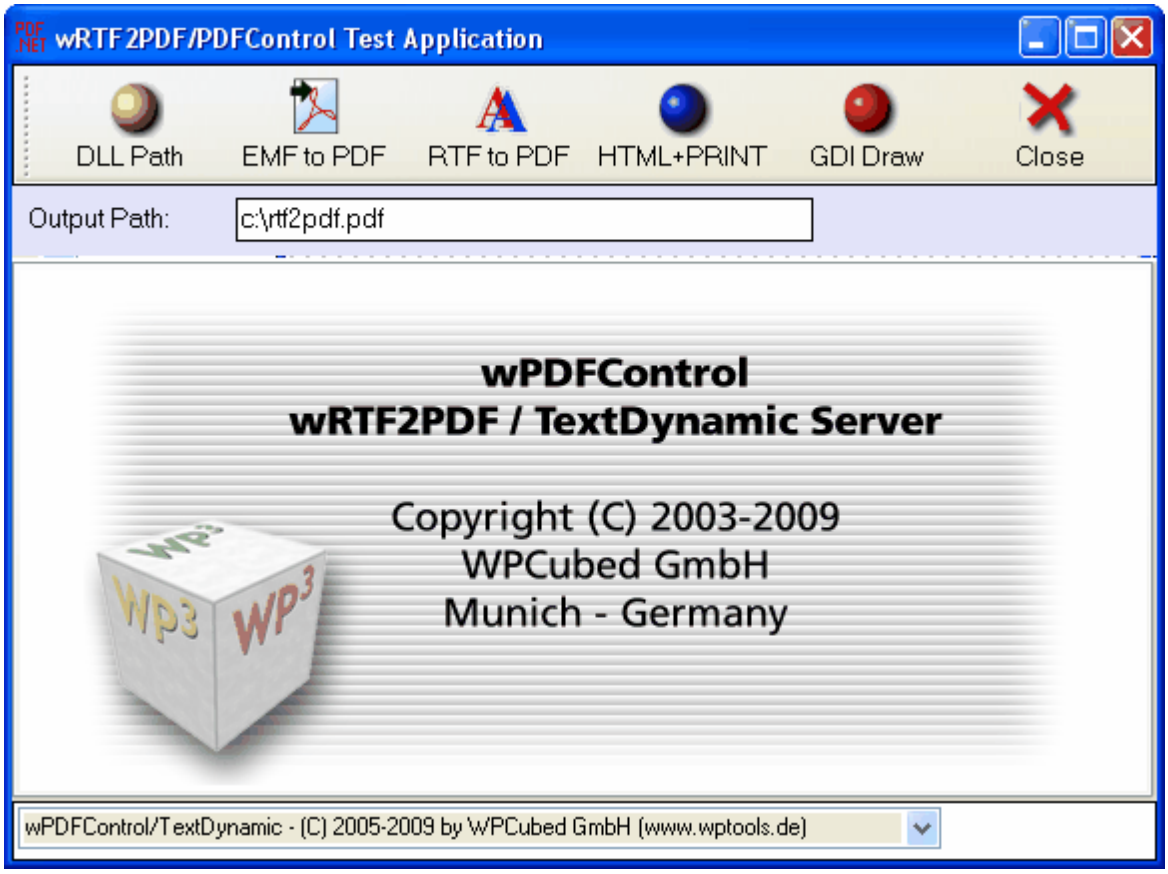
Our component wRTF2PDF/TextDynamic Server offers the same possibility as wPDFControl. It does everything what wPDFControl does but, of course, its main feature is the integrated powerful text rendering engine.

Since wRTF2PDF V3.5 it is also possible to create text and save in RTF or HTML, even MIME format. This is possible since the component exports all of its internal word processing abilities through a set of interfaces. This interfaces are in fact mostly the same as the interfaces used by our visual word processing control "TextDynamic". Only the interface IWPMemo has been replaced by a different interface (IWPEditor) which contains about 70% of the methods originally found in IWPMemo.

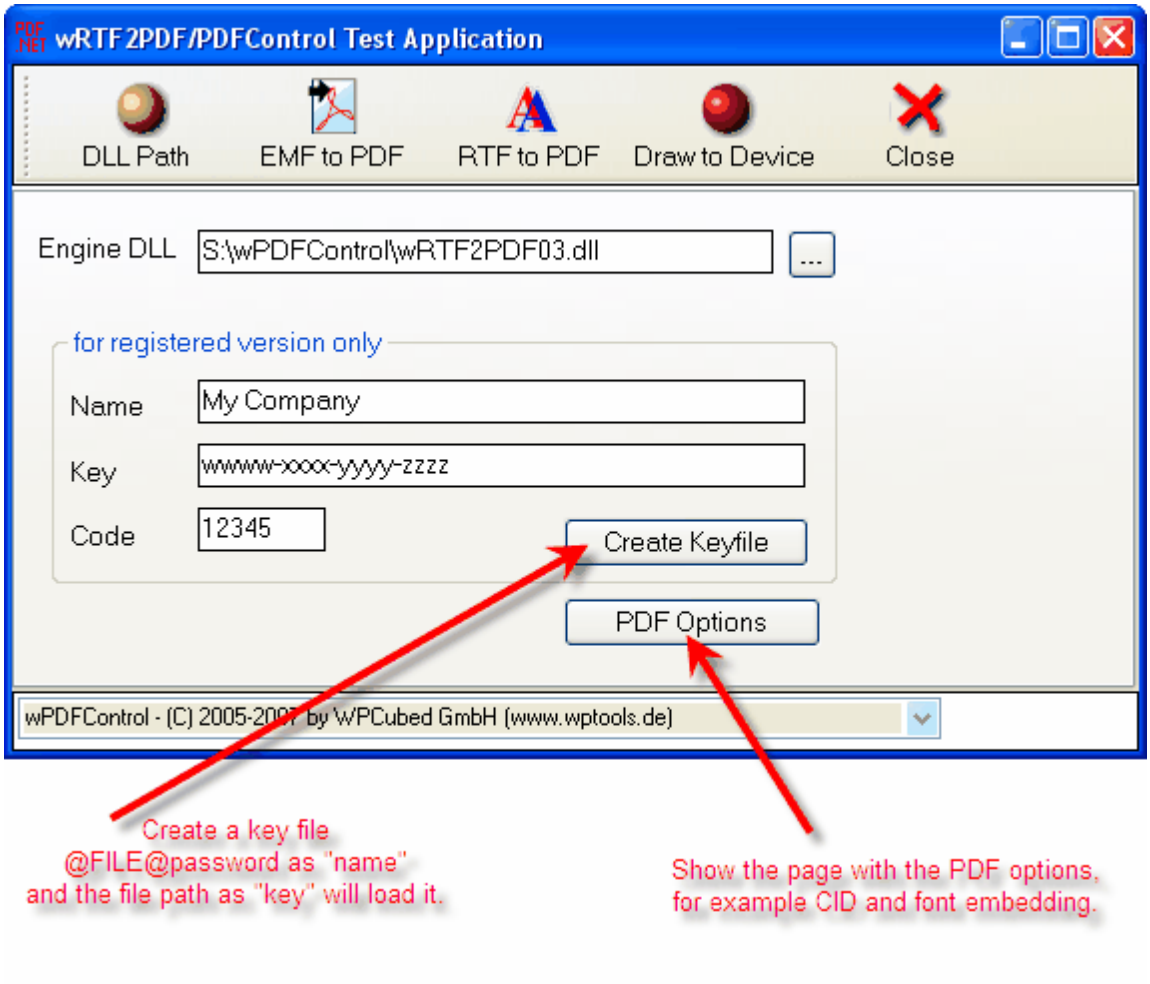
wPDFControl can be used as .NET control, as ActiveX and also as regular DLL. For the DLL we have included a C++ include file. This manual describes the use with .NET.

2.10.1 wPDFControl/RTFtoPDF Demo

The setup installs a demo application which can be used to test the control:



On page 2 you can specify the engine dll and, with registered version, set your license keys

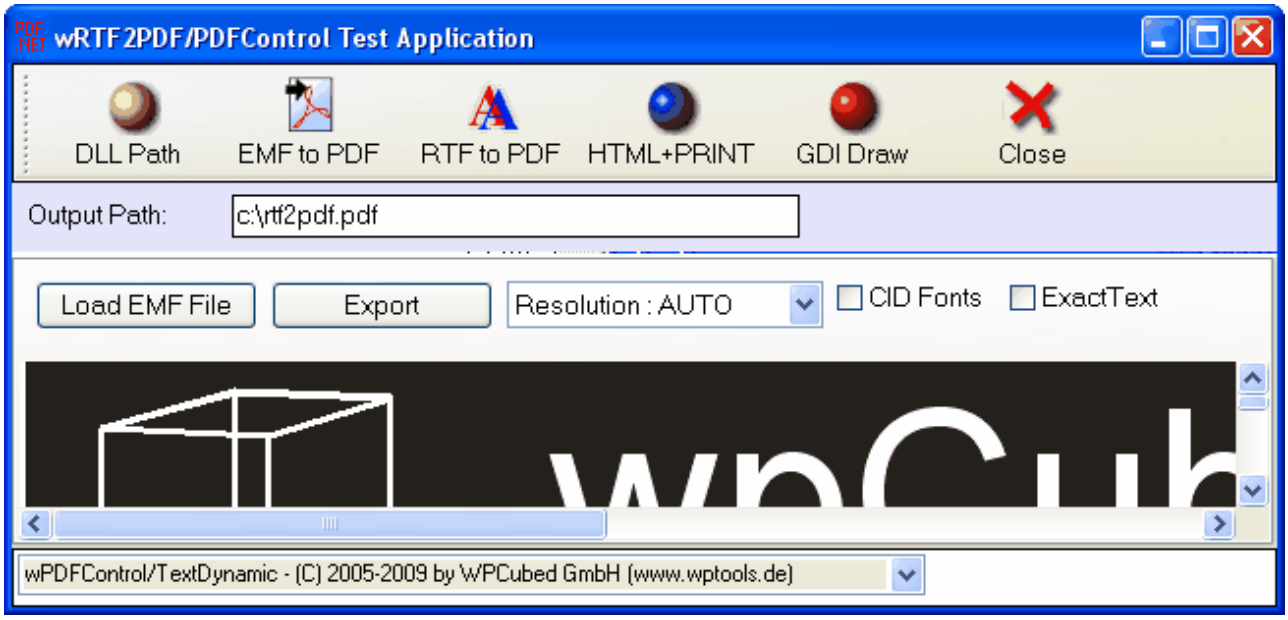


The Options grid is displayed when you click on PDF Options.

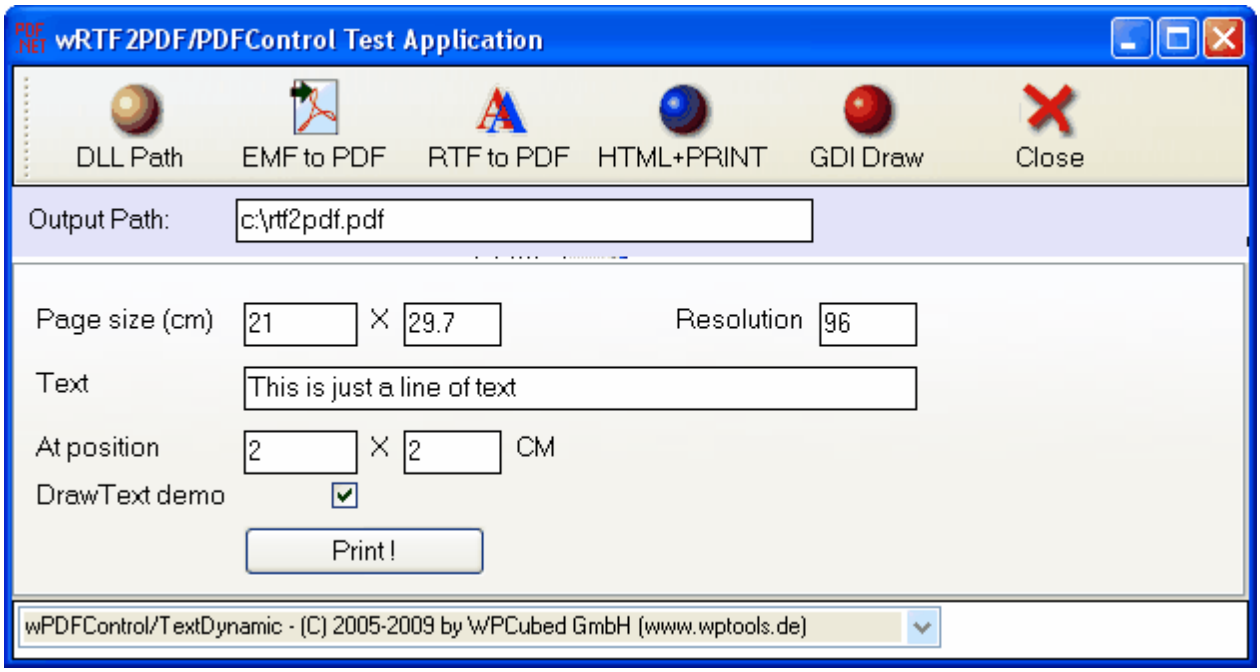
Name	Value
Encode	
Compression	
PageMode	
FontMode	
Encryption	
InputFileMode	
MonoThumbnail	
JPECompress	
Enhanced	
DevMode	

You can specify integer values for certain properties, for example a "1" in the field FontMode will embed all fonts.

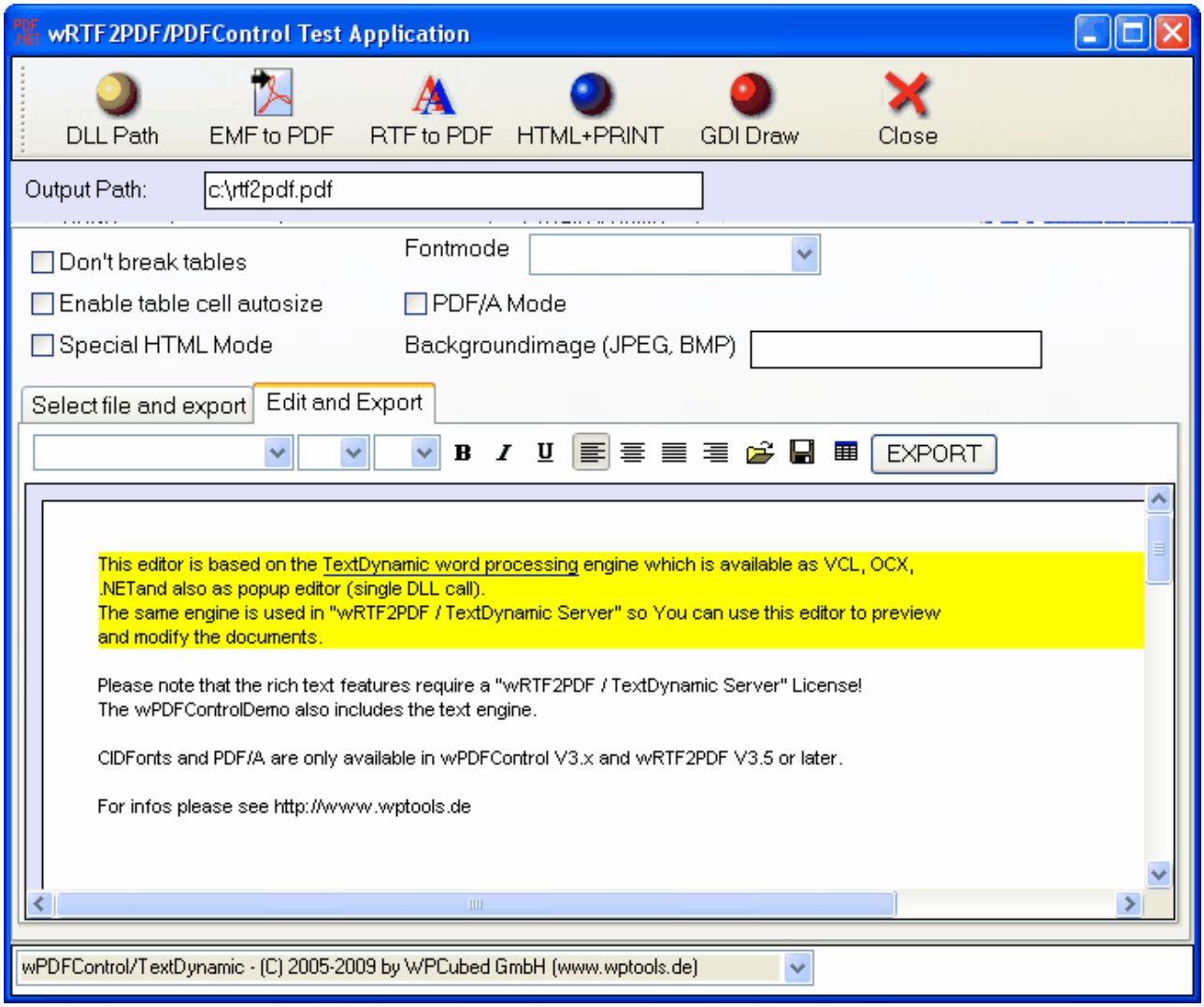
Click on "EMF to PDF" to load an EMF file and export it to PDF



We integrated a simple print to HDC (Device Handle) demo. You can change the resolution and the text.



RTF to PDF conversion is only useable if you use the licensed RTF2PDF DLL or if you use the wPDFControl DEMO version. Here you can export a RTF, HTM or WPT file:



The editor used on this page can be licensed as product TextDynamic (.NET and OCX, C++).

2.10.2 Quick Start - wRTF2PDF / TextDynamic Server

Using wRTF2PDF / TextDynamic Server you can convert RTF and HTML/CSS files to PDF. You can also create new RTF, HTML and PDF files using a powerful API. It is also possible to work with data fields, please see the documentation about the "mail merge" feature.

Installation

Please install the .NET wrapper class for wPDFControl into your .NET development system:

MS Visual Studio: Click right on toolbox and select 'Customize'. Use the 'Locate ...' button to show the file open dialog to add the assembly wPDF.DLL.

Borland C# Builder: Click right on the toolbox and select 'Customize'. Add the path of the assembly wPDF.DLL (3rd page on the dialog).

Please activate the 3 components 'PDFControl, PDFPropDlg, RTF2PDF' which are all in the namespace 'wPDF'.

When you use the included demo projects or create your own project you always need to add a reference to the wPDF assembly installed in the wPDFControl directory under \DLL\.NET\Demo\wPDF.DLL or \DLL\.NET\Full\wPDF.DLL.

Please note that your application will always need

a) wPDF.DLL and
b) either of wPDFControlDemo.DLL or wRTF2PDF0x.DLL in its directory to run.

A) Alternative 1: Create the component in code

```
        {
            wPDF.RTF2PDF rtF2PDF1;

            rtF2PDF1 = new wPDF.RTF2PDF();
            rtF2PDF1.PDFMode = wPDF.eDevMode.ClipRectSupport;
            rtF2PDF1.PDFOptions = wPDF.ePDFOptions.CreateAutoLinks;
            rtF2PDF1.Thumbnails = wPDF.eThumbnails.Color;

            rtF2PDF1.SetLicense("name", "key", 1); // For registered version

            openFileDialog1.Filter = "Text Files (*.TXT;*.RTF;*.HTM)|*.TXT;*.
RTF;*.HTM;*.MHT;*.MSG";

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                rtF2PDF1.Memo.LoadFromFile(openFileDialog1.FileName); // @"c:\a.
rtf");
                rtF2PDF1.PdfCreator.PDFFile = textBox1.Text;
                rtF2PDF1.PdfCreator.Print();
            }

            rtF2PDF1.Dispose();
            rtF2PDF1 = null;
        }
```

B) Alternative 2: Add the component to the form

1) Select the RTF2PDF and click on the form.
2) Add a button to the form to load file and export

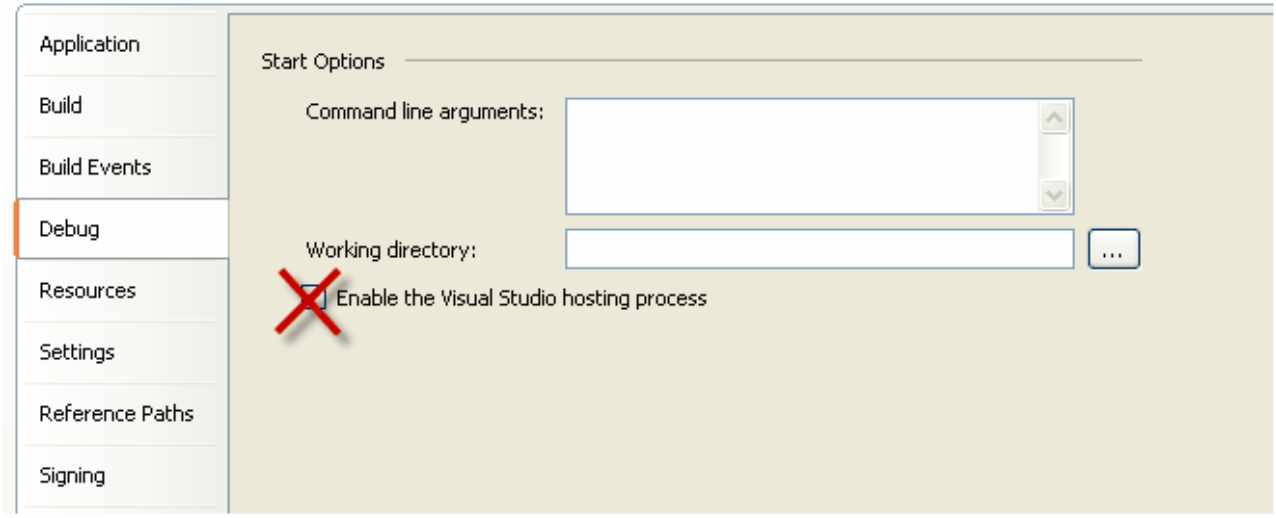
```
private void button1_Click(object sender, System.EventArgs e)
  {
   rtF2PDF1.SetLicense("name", "key", 1); // For registered version
   IWPEditor memo = rtF2PDF1.Memo;
   IWPTextCursor cursor = memo.TextCursor;
   IWPPdfCreator pdf = rtF2PDF1.PdfCreator;

   string path = "..\\..\\";
```

```
  if(memo.LoadFromFile(path+ "demo.rtf", false, ""))
  {
        pdf.PDFFile = path+ "new_pdf.pdf";
        pdf.Print();
  }
 else MessageBox.Show("Cannot load RTF");
}
```

Please see the other examples in the RTF2PDF section ....

C) Troubleshooting

In case You use the wRTF2PDF demo DLL and see the error "Runtime 207" when closing the application please deactivate the Visual Studio 2008 hosting process by removing the check here:



2.10.3 Quick Start - Generic PDF Creation

A) Installation

Please install the .NET wrapper class for wPDFControl into your .NET development system:

MS Visual Studio: Click right on toolbox and select 'Customize'. Use the 'Locate ...' button to show the file open dialog to add the assembly wPDF.DLL.

Borland C# Builder: Click right on the toolbox and select 'Customize'. Add the path of the assembly wPDF.DLL (3rd page on the dialog).

Please activate the 3 components 'PDFControl, PDFPropDlg, RTF2PDF' which are all in the namespace 'wPDF'.

When you use the included demo projects or create your own project you always need to add a reference to the wPDF assembly installed in the wPDFControl directory under \DLL\.NET\Demo\wPDF.DLL or \DLL\.NET\Full\wPDF.DLL.

Please note that your application will always need
a) wPDF.DLL and
b) either of wPDFControlDemo.DLL or wPDFControl0x.DLL or wRTF2PDF0x.DLL in its directory to

run.
Which of this DLLs are required depends on wether you are using the demo, the standard PDFControl or the enhanced RTF2PDF control which includes the PDF functionality with added RTF export functions.

B) Add the components to the form

Select the PDFControl and click on the form.
Select the PDFPropDlg component and click on the form.

Connect both component by selecting the PDFControl instance in the property 'PDFPropDlg.PDFControl'.

C) Add a button to the form to show the important PDF properties

In the OnClick event handler of button1 please use the code:

```
private void button1_Click(object sender, System.EventArgs e)
{
      pdfPropDlg1.Execute(false);
}
```

D) Add code to the form to show a file open dialog and let the user select metafiles and bitmap files which will be exported to PDF.

```
private void button2_Click(object sender, System.EventArgs e)
{
   // We need a filename
   if(pdfControl1.FileName=="") pdfPropDlg1.Execute(false);

   // A dialog to select a file name
   System.Windows.Forms.OpenFileDialog OpenDia =
   new System.Windows.Forms.OpenFileDialog();
   OpenDia.Filter = "Graphic Files|*.WMF;*.EMF;*.BMP;*.JPG;*.JPEG";
   OpenDia.Multiselect = true;
   // If ok then create a PDF file with all selected EMF files
   if(OpenDia.ShowDialog()==System.Windows.Forms.DialogResult.OK)
   {
      pdfControl1.BeginDoc();
      try
      {
        for(int i=0;i<OpenDia.FileNames.Length;i++)
          {
         pdfControl1.DrawImage(OpenDia.FileNames[i],100);
        }
      }
      finally
      {
         pdfControl1.EndDoc();
      }
   }
}
```

E) What you can also do:

Draw using the property 'Canvas' which is compatible to the Graphics class:

```
if(pdfControl1.BeginDoc(filename))
```

```
{
      Font font = new Font("Arial", 10);
      SolidBrush brush = new SolidBrush(Color.Black);
      Pen pen = new Pen(Color.Red, 1);
      pdfControl1.StartPage();
      pdfControl1.Canvas.DrawString("Hello World",
                font, brush,40, 50);
      pdfControl1.Canvas.DrawRectangle(pen, 20,20,300,300);
      pdfControl1.EndPage();
}
```

We have added a "font manager" to make it easier to use fonts. You don't have to create a font object, only to change the property pdfControl1.Font like this:

```
      pdfControl2.Font.Size = 11;
      pdfControl2.Font.Color = Color.Red;
      pdfControl2.TextOut("Hello ",40, 50);
      pdfControl2.Font.Style = FontStyle.Bold;
      pdfControl2.TextOut("PDFCONTROL ");
```

When you pass "\n" a new line will be created by advancing using
the last font height and resetting x to the position defined the last time with TextOut(string,x, y). The intern position pointer will be advanced by using the return value of the MeasureString function.

2.10.4 Tip: Use SaveDC / RestoreDC

Please note - PDF does not offer a possibility to reset the current coordinate transformation. So our PDF engine has to create a complicated conversation vector to reset to default, in case your programming code demands it. This conversion vector seems to make Acrobat 5 trouble if also a rotation is involved.

So we recommend to use SaveDC / RestoreDC, it is available in .NET as Graphics.Save and Graphics.Restore

Example:

```
void DrawSomething(Graphics g, ....)
{
            System.Drawing.Drawing2D.GraphicsState saved = g.Save();//= SaveDC
            try
            {

             ...

                // not required since we use RestoreDC:
                // g.ResetTransform();
                // g.PageUnit = OrigGraphicsUnit;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            finally
            {
                g.Restore(saved);// = RestoreDC
            }
        }
```

```
}
```

SaveDC / RestoreDC is also useful when working with clipping rectangles. It is not required for the Clipping Paths which make their own brackets.

2.10.5 wPDFControl .NET

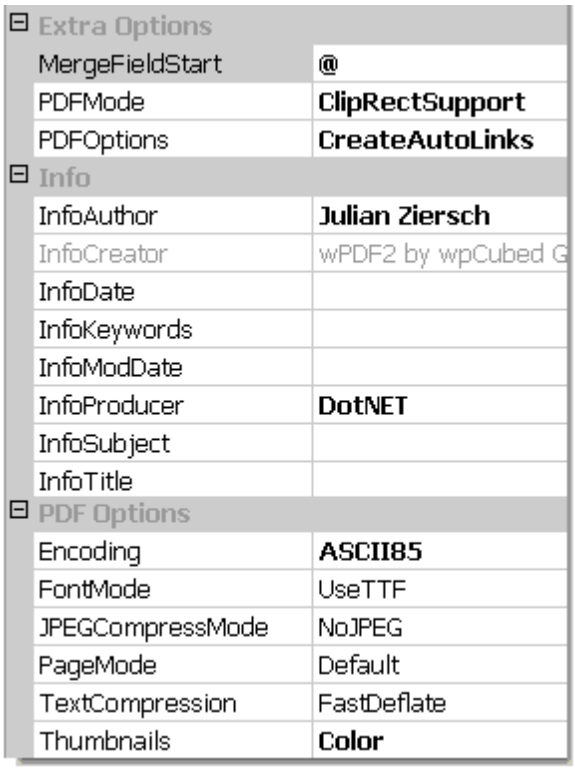
2.10.5.1 PDFControl

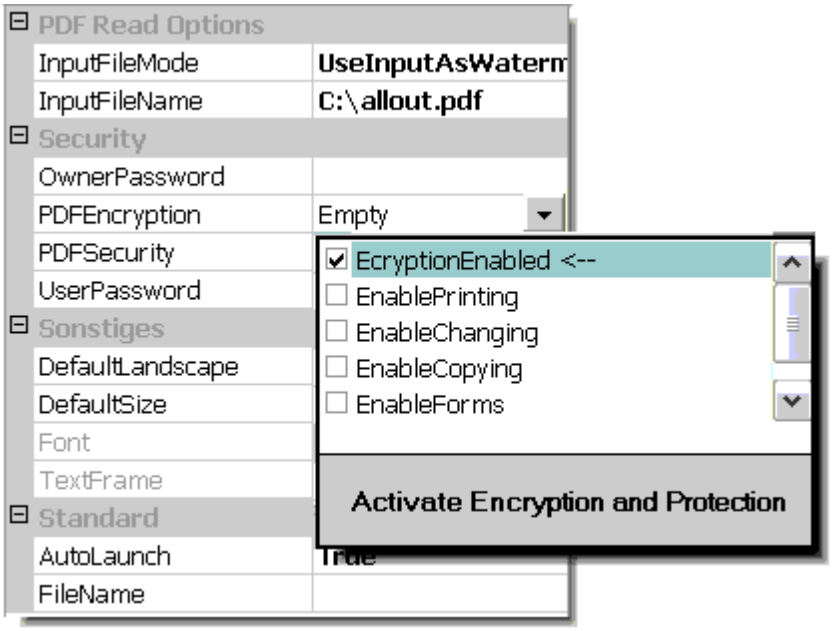
PDFControl ist the main component of PDFControl.NET. It contains a multiude of properties to change the way PDF is created and makes events available to be used in your C# or VB.NET application.

To use it you should specify a file name it its property 'FileName'. This is the file which will be created when you execute BeginDoc().

All properties have descriptive comments. The "flags" properties PDFMode, PDFCoptions and PDFEncryption will show a dropdown editor to make it easy to select the items you need.

PROPERTIES





In addition there is the property **'Canvas'** which is a true 'Graphics' object to be used with your .NET drawing code.

EVENTS

Generic	
OnNamedEvent	**pdfControl2_OnNamedEvent**
Information	
OnError	**pdfControl2_OnError**
OnMessage	**pdfControl2_OnMessage**
MailMerge	
OnGetText	**pdfControl2_OnGetText**
No active page	
AfterBeginDoc	
AfterEndDoc	
AfterEndPage	
BeforeBeginDoc	
BeforeEndDoc	
BeforeStartPage	
PDF page is open	
AfterStartPage	
BeforeEndPage	**pdfControl2_BeforeEndPage**

The event 'OnNamedEvent' will be triggered for all events under 'No active page' and 'PDF page is open'.

Esspecially useful is 'AfterStartPage' to draw a watermark on each page. This can also be used when you export and RTF file using RTF2PDF:

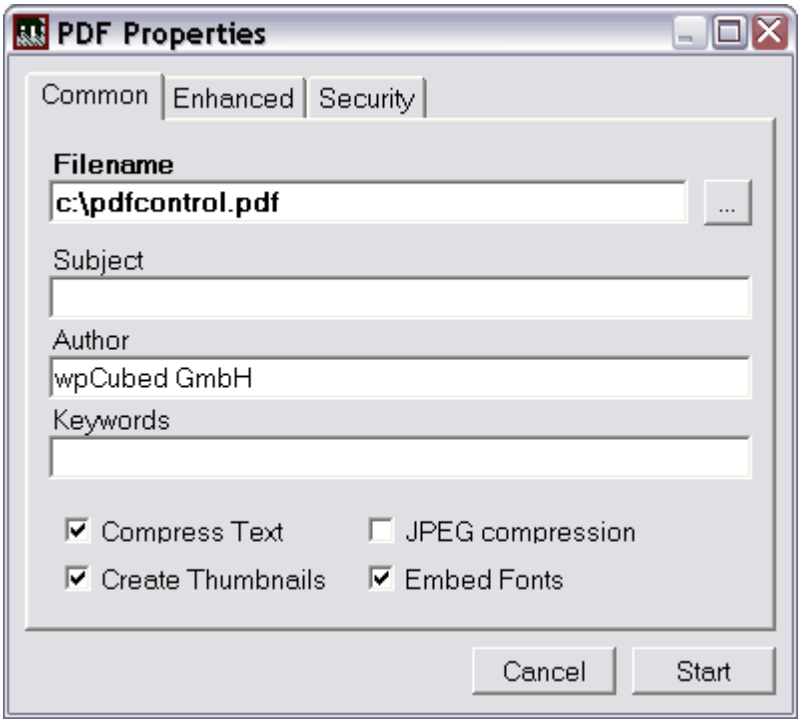
2.10.5.2 PDFPropDlg

This component displays this dialog when the procedure Execute() is called:

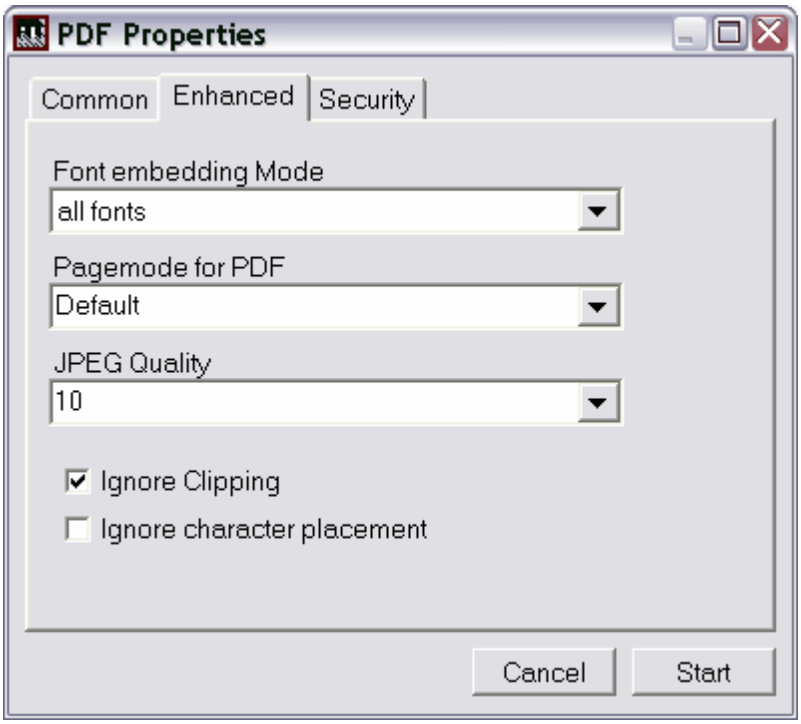
To use it you need to assign its property 'PDFControl' to either an instance of PDFControl or

RTF2PDF.

the first page with some standard properties



the second page with advanced settings



the third page can be used to change the security settings

Before the dialog is displayed the event **BeforeShowDialog** is triggered. You can use it to change the captions and or visibility status of certain elements:

```
private void pdfPropDlg1_BeforeShowDialog(object Sender, wPDF.ConfigPDF Dialog)
{
       Dialog.OKBtn.Text = "Start"; // Change the button text
       Dialog.linklabel.Visible = false; // Hide the URL label
}
```

This dialog does not publish all properties - only those which are likely of interest for the user of your application.

2.10.5.3 Properties

2.10.5.3 A) PDF Options

Property PageMode
By modifying this property you can select how the PDF file has to be displayed when the reader opens it.

Property Thumbnails
If this property is set to 'Monochrome' or 'Color' the PDF export component will create thumbnails in the PDF file.
Please note the the Acrobat(R) Reader Version 5 will create thumbnails automatically if none are found in a PDF document. This thumbnails have a high resolution since they are created using the complete information of a page.

Property TextCompression
By modifying this property you can let the PDF engine compress text. By using compression the file will be reasonable smaller. On the other had compression will create binary data rather than ASCII data. Please note that bitmaps and embedded fonts will be always compressed, either using Deflate or JPEG compression.

Property JPEGCompressMode
wPDF can compress bitmaps using JPEG. This will work only for true color bitmaps (24 bits/pixel) and if you have set the desired quality in this property.

Property Encoding
If data in the PDF file is binary it can be encoded to be ASCII again. Binary data can be either compressed text or graphics. You can select HEX encoding or ASCII95 which is more effective then HEX.

Property FontMode
Using this property you can decide weather TrueType fonts should be embedded in the PDF file or not. If fonts were not embedded in the PDF file text can be displayed wrongly if the used fonts are not installed on the PC of the reader of the PDF file. On the other hand embedded fonts causes the PDF files to be much larger. The size of the usual font file is 150KB! The embedding also slows the creation process down.

You can set wpOnEmbedFonts in property ExtraMessages to get a message (event OnError) once font data is embedded.

UseTrueTypeFonts : use true type fonts but does not embed the font data.

EmbedTrueTypeFonts : embed data of all used fonts. (You can still use ExcludedFonts to specify certain standard fonts)

EmbedSymbolTrueTypeFonts : embed only symbol true type fonts. (such as WingDings, etc.)

UseBase14Type1Fonts : Do not use true type. When this mode is selected you can only use Arial, Courier New and Times New Roman fonts.

Subset Embedding: Since version 2 it is now possible to ***reduce the size of the embedded data*** by removing the description for unused characters.

There are 2 options which do this for you:

EmbedSubsetCharsets will embed all the characters which are used in the codepage you selected.

EmbedSubsetUsedChar includes only those characters which have been used and so produces the smallest files.

2.10.5.3 B) Extra Options

PDFOptions

CreateAutoLinks - if active the PDF engine will create web links over text which starts with http://. This makes it easy to create a link to a web page.

```
Canvas.TextOut("http://www.wptools.de",10,20);
```

Please note that the autolink feature requires that the complete link is printed at once. Links which are inside longer text strings are not recognized.
Note: You can always create links using the Hyperlink procedure.

PDFMode

makes it possible to change the way the drawing commands are interpreted. For standard output this property should remain unchanged but if you create the output in your application its use might be required.

NoBITBLTFillRect: BitBlt is not used to draw a filled rectangle.

WhiteBrushIsTransparent: Do not fill objects which use a white background. Please note that this has no effect on bitmaps, only on rectangles or text which is printed opaque.

ExactTextPositioning: In general the PDF engine draws text to not only match the input but also look good. If your application requires that each character has to be at the same position as in the input you can use this flag to get a 1:1 output. (Implementation note: In the PDF file an efficient multi string output command is used, not several single text drawing commands)

NoTextRectClipping: Text can be clipped to its bounding box. To switch this off you can use this flag. It is a good idea to use this flag if your application uses a lot of TextRect() without expecting clipping.

ClipRectSupport: IntersectClipRect() and SelectClipRgn() are supported.

DontStackWorldModifications: If your application uses the windows function SetWindowOrgEx () a lot please set this flag. Normally please do not use this flag. It is only required if you see a small, 1 pixel shift in vertical or horizontal lines. If you have access to the printing code you can add SaveDC/RestoreDC around blocks which work with a different origin or resolution to avoid this small visual problem in the created PDF file. Normally you should not use this option!

NoTextRectClipping: Use the windows text output API you can provide a rectangle which is used to clip you are just printing. This is useful if you are printing table cells and don't want to overprint the contents of the neighbor cell. Usually this clipping rectangle is not active and wPDF will not add unnecessary clipping - but if your application uses this clipping without a real need for it, you can improve the quality of the PDF file by switching this flag on. This will skip the creation of clipping rectangles for text output commands.

DontAdjustTextSpacing: wPDF will normally use the character and word spacing to render the text to match the width requirement set by windows. This is necessary because the fonts in PDF have a slightly different width than calculated by windows. The reason are rounding errors in windows (or visual optimation) which works with integer positions while PDF uses floating points.
If you use this flag wPDF will not try to enlarge or shrink the text. Normally you won't see a problem, except that you are printing text lines which consist of different parts (font styles fro example).
This flag has no effect if wpExactTextPoitioning is active.

DontCropBitmaps: Using the command StretchDIB you can specify a rectangle of the source bitmap which will be printed in the destination rectangle. Unless this flag is active the PDF engine will internally remove the unprinted data.

DontCropBitmaps: The PDF engine will crop images if they are only partly painted. You can use this flag to switch this mode off.

AllowTransparentBit: Monochrome bitmaps can be optionally painted transparently if they are painted using an OR "ROP" mode or if the flag wpWhiteBrushIsTransparent. This will set the PDF format version to V1.3.

DevMode

This property controls the PDF engine. You can use SetIProp(10, value) to update this property.

bit 1: NoBITBLTFillRect
bit 2: NeverFill
bit 3: WhiteBrushIsTransparent
bit 4: ExactTextPositioning
bit 5: NoTextRectClipping
bit 6: ClipRectSupport
bit 7: DontStackWorldModifications
bit 8: DontAdjustTextSpacing
bit 9: DontCropBitmaps
bit 10: AllowTransparentBit
bit 11: AlwaysHighResPDF
bit 12. NeverHighResPDF
bit 13: UseFontWidthArgument

bit 14: NoTextScaling
bit 15: MetaIsDOTNETEMF (internally used)
bit 16: InputIsWMFData

2.10.5.3 C) Security

Property PDFEncryption

EncryptFile - to switch on encryption
EnablePrinting - allow the user to print the PDF file
EnableChanging - allow to change the PDF file
EnableCopying - allow to copy text from the PDF file
EnableForms - enable interactive elements (do not yet apply)
LowQualityPrintOnly - if you use high compression this flag lets you switch off high resolution printing.

The property changes the rights of the user and if the file is encrypted..
You can protected the file from viewing if you also set the property UserPassword or you can simply prohibit copying. Currently wPDFControl supports the 40-bit and 128 bit PDF encryption. The later also causes the PDF to use PDF version 1.4.

property Security (Std40bit, High128bit)
You can change between

Property OwnerPassword
The password is required to edit an encrypted PDF file. If you don't set a password here a password will be randomly created when you enable Encryption.

Property UserPassword
This is the password which will be used to encrypt the file. If you don't provide a password the rights for the user can be still restricted using 'Encryption'. But the user will then not be prompted for a password.

2.10.5.3 D) PDF Info

This properties change the fields which are used for the document info of the created PDF file. This information can be viewed in the PDF reader under 'Document Properties'. It is not printed.

2.10.5.3 E) PDFAMode

This property enables PDF/A support:

Values:
0=off
1=enabled

PDF/A is a new norm which based on PDF 1.4 - it was created to provide a guildline for the creation of document files which stay readable for the time to come. So the major demand for PDF/A complient files is that the used font files are embedded. Security measures are forbidden in PDF/A complient files as are links to external files. But there is more to PDF/A. We have checked the component carefully against the final documentation of PDF/A.

When you use RTF2PDF V3 additional (invisible) tags will be added to the PDF data. This tags make it possible to identify layout elements (such as header or footer texts) on a page. They can be also used by a PDF reader to convert the PDF data into text paragraphs, something which is otherwise at least difficult and impossible if a paragraph spans a page. The wPDF engine will also create tags to mark table cells. wPDFControl3 will also add the document

information as XMP data to the PDF file.

PDF/A support is a new feature in wPDFControl V3.

2.10.5.3 F) CidFonts

This property enables the support for text written used by Character Identifiers. Here in the PDF text numbers are written which are mapped to certain glyphs in an embedded font. A special additional mapping table makes sure that the text can be extracted as unicode text. When the CID feature is used this means the fonts are embedded as subsets in a highly efficient way. PDF files become smaller this way. Since it works with character ids and not with charsets the export for say, Russian text, also works without having specified the charset explicitly. wPDF uses as character IDs the UNICODE values of each character - it can have used any number but we wanted to preserve the most information of the source text in the PDF as possible. Please note that the support for asian languages does NOT use the CID feature. Asian languages use special, perdefined fonts and mapping tables and require the charyset to be known to be propperly exported.

CidFontMode Values:

0 - CID feature is not used. The property FontMode rules font embedding
1 - all fonts are embedded. Unicode values will be used as character ids
2 - only symbol fonts will be embedded

Cid Font support is a new feature in wPDFControl V3.

2.10.5.4 RTF to PDF

2.10.5.4 A) RTF2PDF Version 2 API

Unless you intend to use wRTF2PDF without the COM interfaces please stop reading here and see the dedicated RTF2PDF reference manual.

The RTF2PDF component inherits all properties and methods of PDFControl.

It adds methods to load, manage and export RTF files to PDF.

To convert a RTF file you first load it and then export it.

```
System.Windows.Forms.OpenFileDialog OpenDia =
new System.Windows.Forms.OpenFileDialog();
OpenDia.Filter = "RTF Files|*.RTF";
if(OpenDia.ShowDialog()==
     System.Windows.Forms.DialogResult.OK)
{
   rtF2PDF1.LoadRTF(OpenDia.FileName);
   rtF2PDF1.Export("C:\\Test.PDF");
}
```

```
//Loads a RTF file
public void LoadRTF(String FileName)

//Loads RTF from a string
public void LoadRTFString(String RTFFormat)

//Appends RTF from a string
public void AppendRTFString(String RTFFormat)
```

```
//Saves the RTF file
public void SaveRTF(String FileName)

//Appends RTF file to the data stored in the RTF Engine
public void AppendRTF(String FileName)

//Converts <fieldnames> in the loaded RTF data into the special fields
//(insertpoints) which can be used for mailmerge.
public void MergePrepare()

//Starts the mailmerge process. Please note that you can execute this,
// write a PDF file and execute it again without having to reload the RTF data!
//The data is retrieved using the OnGetText event procedure.
public void MergeExec()

//writes the RTF data to an intern data storage. To restore it use the procedure
MergeRestore.
public void MergeBackup()

//Restores text saved with MergeBackup()
public void MergeRestore()

//Define how many pages should be printed on one PDF page side by side.
//The standard is 1 to print one RTF page on one PDF page.
// "2" would print 2 RTF pages on one PDF pages and also double the width of the PDF page.
public void SetColumns(int Value)

// Start the RTF to PDF conversion. During the conversion process the
// callbacks are executed to let you fill in graphics if you need to.
// This makes it possible to create a watermark on each page.
// It uses the 'Filename' property as output name
public void Export()

// Start the RTF to PDF conversion. During the conversion process the
// callbacks are executed to let you fill in graphics if you need to.
// This makes it possible to create a watermark on each page.
public void Export(String OutputFileName)

//Sets the zoom value for the RTF to PDF conversion.
// It can be used to shrink or enlarge the text.
// To print 2 A4 pages on one A3 page use columns=2, Rotation=90, zoom=100.
// To print 2 A4 pages on one A4 page use zoom=50!
public void SetZoom(int Value)

//Changes the rotation of the PDF page.  To print landscape use the value 90.
public void SetRotation(int Value)
```

2.10.5.4 B) RTF2PDF Command IDs

The internal RTF engine is controlled by the function **ExecComand()** with the command numbers listed here.

The ActiveX wraps this function as methods as ExecCommand, ExecIntCommand and ExecStrCommand.
Normally you will use ExecStrCommand() since this method lets you specify a string as a parameter.

C example code to create a PDF engine, load a file and print it to PDF.

```
pdf=wpdfInitializeEx(&Info);                    // Initialize the PDF Engine
if(wpdfBeginDoc(pdf,"C:\\aTestPDF.pdf",0))        // Create PDF File
```

```
{
      wpdfExecCommand(pdf,1000,0,"",0);              // Initialize the RTF engine
      wpdfExecCommand(pdf,1002,0,"C:\\test.RTF",0);  // Load a RTF file
      wpdfExecCommand(pdf,1100,0,"",0);              // Export this RTF file to PDF
      wpdfEndDoc(pdf);
}
wpdfFinalize(pdf);
```

You can use the header file wPDF.H and wpdf_class.h for convenient binding.

Tip: Please also see: Commands to **print text** (to printer, not PDF!)

Example VB.NET Code to convert a RTF file to PDF:

```
       If PDF.StartEngine(DLLNAME.Text, "", "", 0) Then ' you your license info here !
            If PDF.BeginDoc(PDFNAME.Text, 0) > 0 Then ' Start a PDF document
                PDF.ExecIntCommand(1000, 0) ' Initialize the RTF engine
               PDF.ExecIntCommand(1024, 1) ' use the printer as reference (suggested)
                PDF.ExecStrCommand(1002, RTFNAME.Text) ' Load a RTF file
                PDF.ExecIntCommand(1100, 0) ' Export this RTF file to PDF
                PDF.EndDoc() ' Close the PDF document
            End If
        End If
```

Example VBS Code to convert RTF to PDF: uses the ActiveX)

```
    Set PDF =  CreateObject("wPDF_X01.PDFControl")
    PDF.INFO_Date = Now
    PDF.INFO_Author = "Julian Ziersch"
    PDF.INFO_Subject = "Test file"
    ' ... set other properties

    PDF.StartEngine "c:\wPDFControl\DLL\wRTF2PDF01.dll", "LIC_NAME", "LIC_KEY" ,0 ' License
    Info!
    PDF.BeginDoc "Test.PDF", 0
    PDF.ExecIntCommand 1000, 0
    PDF.ExecIntCommand 1024, 1
    PDF.ExecStrCommand 1002, "Demo.RTF"
    PDF.ExecIntCommand 1100, 0
    PDF.EndDoc
    PDF.StopEngine
    Set PDF = Nothing
```

? WPCOM_RTFINIT = 1000

Initializes the RTF engine. Must be used right after **Initialize**().
Only if it was used the other commands may be used!

? WPCOM_RTFVersion = 1001

The return value of the routine WPDF_ExecCommand is the version number of the RTF engine.

Tip: You can use WPCOM_RTFVersion to check if the engine understands RTF commands. I
if it does not the return code is 0.

? WPCOM_RTFLOAD = 1002

Loads the RTF file specified by the provided name.
All other parameters are ignored.

? WPCOM_RTFAPPEND = 1003

Appends the RTF file specified by the provided name.
All other parameters are ignored.

? WPCOM_RTFMAKEFIELDS= 1004

Converts <fieldnames> in the loaded RTF data into the special fields (insertpoints) which can be used for mailmerge. .

? WPCOM_RTFMERGE = 1005

Starts the mailmerge process. Please note that you can execute this, write a PDF file and execute it again without having to reload the RTF data! The data is retrieved using the OnGetText callback/event procedure.

? WPCOM_RTFSAVE = 1006

Saves the RTF file to the filename specified as parameter.

? WPCOM_RTFBACKUP = 1007

WPCOM_RTFBACKUP writes the RTF data to an intern data storage.
To restore it use the command WPCOM_RTFRESTORE.

? WPCOM_RTFRESTORE = 1008

WPCOM_RTFBACKUP writes the RTF data to an intern data storage.
To restore it use the command WPCOM_RTFRESTORE.

? WPCOM_RTF_PAGEWIDTH = 1010

Set the page-width of the current RTF file to the width specified in the parameter value as twips (1/1440 inch).

? WPCOM_RTF_PAGEHEIGHT = 1011

Set the page-height of the current RTF file to the height specified in the parameter value as twips (1/1440 inch).

? WPCOM_RTF_MARGINLEFT = 1012

Set the left margin of the current RTF file to 'value'.

? WPCOM_RTF_MARGINRIGHT = 1013

Set the right margin of the current RTF file to 'value'.

? WPCOM_RTF_MARGINTOP = 1014

Set the top margin of the current RTF file to 'value'.

? WPCOM_RTF_MARGINBOTTOM = 1015

Set the bottom margin of the current RTF file to 'value'.

? WPCOM_RTF_PAGECOLUMNS = 1020

Define how many pages should be printed on one PDF page side by side.

The standard is 1 to print one RTF page on one PDF page. 2 would print 2 RTF pages on one PDF pages and also double the width of the PDF page.

? WPCOM_RTF_PAGEROTATION = 1021

Changes the rotation of the PDF page.
To print landscape use the value 90.

? WPCOM_RTF_PAGEZOOM = 1022

Sets the zoom value for the RTF to PDF conversion. It can be used to shrink or enlarge the text.

To print 2 A4 pages on one A3 page use columns=2, Rotation=90, zoom=100. To print 2 A4 pages on one A4 page use zoom=50!

? WPCOM_RTF_READEROPTIONS = 1023

This command changes a few options for the RTF reader:

Bit 0: all borders which use the width 0 should be invisible
Bit 1: the file is loaded in landscape mode
Bit 2: the page width and page height is swapped at load time

? WPCOM_RTF_USE_PRINTER = 1024

Here you can switch on and off the use of the current printer driver for measuring the text. Since the printer has a higher resolution this causes a better output (less rounding errors) but it will require that a printer is installed.

Value=1 switches the mode on
Value=1 switches the mode off (default)

*(This command has been added to V1.08 - **it is obsolete in V3!**)*

? WPCOM_RTF_PAGESMALLCOLUMNS = 1025

Define how many pages should be printed on one PDF page side by side.

The standard is 1 to print one RTF page on one PDF page. 2 would print 2 RTF pages on one PDF pages and, in contrast to WPCOM_RTF_PAGECOLUMNS devide the page width by the count of columns.

? WPCOM_RTFPRINT = 1100

Start the RTF to PDF conversion.
During the conversion process the callbacks are executed to let you fill in graphics if you need to.
This makes it possible to create a watermark on each page.

? WPCOM_SAVE_PAGE_METAFILE = 1311

Save the page with number param to the file name specified as character buffer.

? WPCOM_SELECT_HTML_MODE = 1312

Enable the special HTML mode with param=1 and switch it off with param=0

? Commands to print text (to printer, not PDF!)

RTF2PDF / TextDynamic Server can also print text to a printer driver.

When using the COM interfaces you can use this commands to select the printing.

Memo.TextCommandStr ID 10 - Select Printer

Memo.TextCommandStr ID 11 - Print Text
Memo.TextCommandStr ID 12 - Get Printernames
Memo.TextCommandStr ID 13 - BeginPrint
Memo.TextCommandStr ID 14 - EndPrint

Please note, printing is not thread save!

When you do not use the COM interface use this command ids instead:

```
WPCOM_SELECT_PRINTER = 1320;  // param = printer name

WPCOM_PRINT  = 1321;          // param = page list

WPCOM_PRINT2 = 1322;          // print memo 2, param = page list

WPCOM_BEGIN_PRINT = 1323;     // when printing multiple documents into one printing cue
WPCOM_END_PRINT = 1324;       // use beginprint/endprint
```

? KeepN and AutoSize Tables

WPCOM_RTF_SUPPORTKEEPN = 1026

The keep control is used to avoid page breaks between paragraphs. The engine will also try to not break up a table which uses KeepN in the first table row.

The parameter 1 switches KeepN support on (default), 0 switches it off

WPCOM_RTF_DONTBREAKTABLES = 1027

With parameter 1 the engine will try to not break up any tables. (default = off)

WPCOM_RTF_NOWIDOWORPHANPARS = 1028

1 activates the orphan/widow control - default is off.

WPCOM_RTF_DisableAutosizeTables = 1029

Activate/Deactivate auto table width calculation - default is off.

Note: to change this properties using the COM interfaces use the Memo.SetBProp API.

? Use COM interfaces with C

In case you intend to use the new COM interfaces with standard C this is possible, too.

Please use the TLB import of your development tool and execute it with the wPDF01.OCX.

We have added this methods to the wrapper class to provide you with IWPEditor, IWPAttrInterface, IWPPDFCreator and IWPReport interfaces.

```
void *CPDFExport::_Memo()
{
    return (void *)wpdfExecCommand(pdfenv,1301, 0,0,0);
}

void *CPDFExport::_Memo2()
{
    return (void *)wpdfExecCommand(pdfenv,1302, 0,0,0);
}
```

```
void *CPDFExport::_AttrHelper()
{
    return (void *)wpdfExecCommand(pdfenv,1307, 0,0,0);
}

void *CPDFExport::_PDFCreator()
{
    return (void *)wpdfExecCommand(pdfenv,1303, 0,0,0);
}

void *CPDFExport::_Report()
{
    return (void *)wpdfExecCommand(pdfenv,1304, 0,0,0);
}
```

In case you have difficulties to import the TLB please let us know.

2.10.5.4 C) Example

Code to convert a RTF file to PDF

Example VB.NET

```
If PDF.StartEngine(DLLNAME.Text, "", "", 0) Then ' you your license info here !
            If PDF.BeginDoc(PDFNAME.Text, 0) > 0 Then ' Start a PDF document
                PDF.ExecIntCommand(1000, 0) ' Initialize the RTF engine
                PDF.ExecStrCommand(1002, RTFNAME.Text) ' Load a RTF file
                PDF.ExecIntCommand(1100, 0) ' Export this RTF file to PDF
                PDF.EndDoc() ' Close the PDF document
            End If
        End If
```

Example C#

```
   System.Windows.Forms.OpenFileDialog OpenDia =
   new System.Windows.Forms.OpenFileDialog();
   OpenDia.Filter = "RTF Files|*.RTF";
   if(OpenDia.ShowDialog()==System.Windows.Forms.DialogResult.OK)
   {
      rtF2PDF1.LoadRTF(OpenDia.FileName);
      rtF2PDF1.Export("C:\\Test.PDF");
   }
```

2.10.5.5 Methods

2.10.5.5 A) Initialisation

Creating metafile with wPDFControl is as easy as 1 - 2 - 3 or better as

BeginDoc - DrawImage - EndDoc.

If you need a a "Graphics" object to draw to, you need to open a 'page' first. To open a PDF page use StartPage.

BeginDoc - StartPage - Canvas.DrawString(..) - EndPage - EndDoc.

Note: Using 'CloseCanvas' you can always flush the graphic output stored in the Canvas property to the PDF file.

To use registered version of PDFControl and RTF2PDF you need to execute the function **SetLicense(String Name, String Code, uint Number)**

to pass your license name, the code and the number. This activates the PDF engine.

Note:

Version 2 uses a license key similar to

(a) StartEngine("somename", "xxxxyyyy", 123456);

Version 3 can use a key such as

(b) StartEngine("somename", "xxxxyyyy**@**zzzzz", 123456);

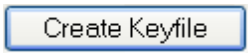
(note the @ sign)

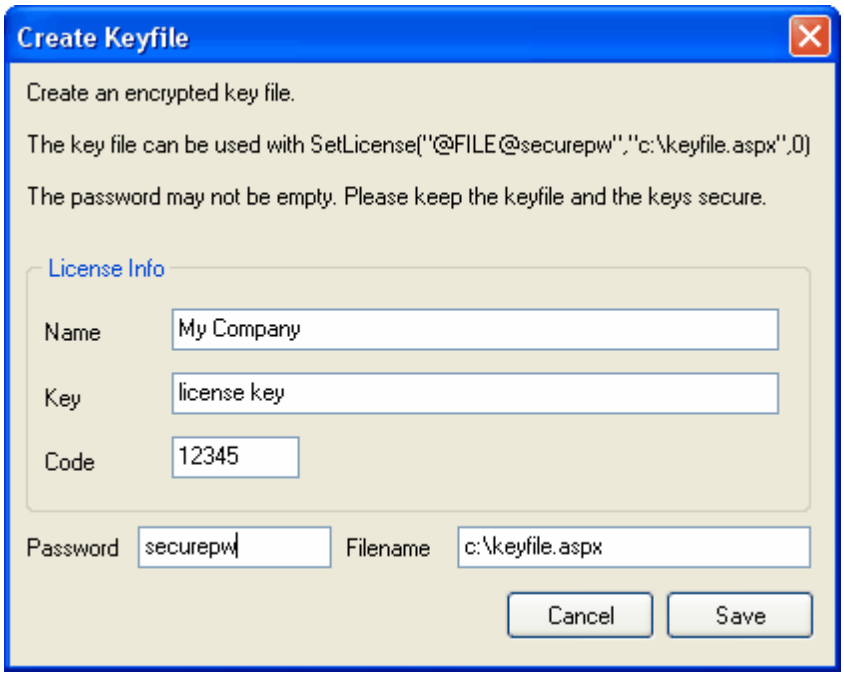
or this

(c) StartEngine("somename", "WWW-XXXX-YYYY-ZZZZ", 123456);

RTF2PDF V3.5 and later will only work with the key using schema (c)!

New: You can use the demo application to create a key file:





This key file can then be loaded using SetLicense ("@FILE@securepw","c:\keyfile.aspx",0)

Note: You can also create a PDF document in a Stream object: Simply pass the a Stream instance to the function BeginDoc.

Overview:

BeginDoc/EndDoc

```
// Use this function to set your license information
public bool SetLicense(String Name, String Code, uint Number)

// Use this function to start a new PDF file
// using the filename set in property FileName
public bool BeginDoc()

// Use this function to start a new PDF file using give filename
public bool BeginDoc(String FileName)

// Use this function to start a new PDF stream (not a file)
public bool BeginDoc(Stream OutStream)

// EndDoc closes a PDF file which has been opened with BeginDoc
public void EndDoc()
```

StartPage/EndPage

```
// StartPage starts a new page in a PDF file opened with BeginDoc.
// Y and Y are measured pt, this is 1 inch / 72
public bool StartPage(int w, int h, bool landscape)

// Starts a page to make that image exactly fit. You can sepcify a zooming value

public bool StartPage(Image image, int ZoomValue)

// Starts a page with a certain ePage
// format: Letter, Legal, Executive, DinA3, DinA4, DinA5 or DinA6
public bool StartPage(ePage format, bool landscape)

// Starts a page using the DefaultPageSize
public bool StartPage()

// Closes a PDF page and writes it
public void EndPage()
```

StartWatermark/EndWatermark

```
// Starts a watermark with a certain name
// and  ePage format: Letter, Legal, Executive, DinA3, DinA4, DinA5 or DinA6
public bool StartWatermark(String Name,ePage format)

// Starts a watermark with the DafaultPageSize and a certain name
public bool StartWatermark(String Name)

/ Closes a PDF watermark and writes it
public void EndWatermark()
```

2.10.5.5 B) DrawWatermark

The PDF engine supports watermarks. The watermarks can be created once and used on as many pages as you like.

The DLL can even import PDF data and convert the imported pages into watermarks which are

names "inpageN" (with N = 1..count of pages). With the function DrawWatermark you can tell the PDF engine to use one of the stored watermarks. You simply have to pass the name of it. If you want to rotate the watermarks to any degree you can do so. But please note that a PDF file with a rotated watermark will not print on all printers. Especially postscript printers seem to not like this feature.

Important is also that on a landscape page the watermark will be also landscape. This means that the watermark has to be portrait before you use it. It will automatically be rotated by using it on a landscape page.

To create a watermark use the function StartWatermark.

To draw a watermark use

```
public void DrawWatermark(String Name, int Rotation)
```

Rotation can be 0, 90 or 180.

2.10.5.5 C) Image Output

wPDFControl provides the function **DrawImage()** with several alternatives to draw a metafile or a bitmap.

If no page has been opened with StartPage a page will be automatically opened in either the DefaultPageSIze, or, if ZoomValue is not 0 to match the (zoomed) size of the image or metafile.

Possible Parameters:

image : This is the Image instance with a bitmap or metafile
FileName: an image file on disc
x,y,w,h : This is the location the image should be drawn on the PDF page - measure in pt (1/72 inch)
ZoomValue: This value (if not 0) can be used to resize the image.
This is useful if the image should be drawn in correct aspect ratio

```
public int DrawImage(String FileName)
public int DrawImage(String FileName, int ZoomValue)
public int DrawImage(String FileName, int x, int y, int w, int h)
public int DrawImage(Image image)
public int DrawImage(Image image, int ZoomValue)
public int DrawImage(Image image, int x, int y, int w, int h)
```

Speciality for metafiles only: - DrawMetafile let you specify the x and y resolution which was used to create the metafile.
This can be useful if the PDFEngine does not pick up the stretching value correctly. (This function implies the ZoomValue set to 100)

```
public void DrawMetafile(Metafile metafile, int xres, int yres)
```

2.10.5.5 D) Hyperlinks

Hyperlinks mark a certain position on the PDF page which can be clicked by the user to either open a certain website or jump to a poition in the PDF file which has been or will be marked with a "BookMark".

To create a link you can use

```
public void Hyperlink(int x, int y, int w, int h, String BookMark)
```

or

public void Hyperlink(Rectangle Rect, String BookMark)

Please note that both functions are using the current Canvas coordinate trasnformation - this means you can use the same coordinates you use for DrawString(), Line() etc.

To create a weblink the bookmark must start with "http://"
To create a link to a certain file it must be start with "Launch://"

2.10.5.5 E) Bookmarks

Bookmarks can be used to mark link destinations in a PDF file.

To create a bookmark you can use

public void Bookmark(int x, int y, String BookMark)

or

public void Bookmark(Rectangle Rect, String BookMark)

Please note that both functions are using the current Canvas coordinate trasnformation - this means you can use the same coordinates you use for DrawString(), Line() etc.

2.10.5.5 F) Outlines

Outlines are displayed in a seperate window by Acrobat Destiller and a good way to navigate a document.

To create an entry use

Create an Outline level as jump to a certain bookmark
public void Outline(String Text, String Bookmark)

Create an Outline level as jump to the current page
public void Outline(String Text)

Create an Outline level as jump to a certain location on the page measured in pt
public void Outline(int x, int y, String Text)

Create an Outline level as jump to a certain location on the page measured in pt
public void Outline(Rectangle Rect, String Text)

To build the outline tree use

public void OutlineChild() // one level down
public void OutlineParent() // one level up
public void OutlineParent(int n) // n levels up

Example:

```
pdfControl2.Outline("Level 1");
pdfControl2.OutlineChild(); // Child on Level 2
pdfControl2.Outline("Level 2 a");
pdfControl2.Outline("Level 2 b");
pdfControl2.OutlineChild(); // Child on Level 3
pdfControl2.Outline("Level 3 a");
pdfControl2.OutlineParent(2); // Go 2 Levels up
pdfControl2.Outline("Level 1");
```

2.10.5.5 G) using "Graphics Canvas"

PDFControl exportes metafiles to PDF.

The wrapper class PDFControl and RTF2PDF also includes a object 'Canvas' which inherits of the class 'Graphics'. This means you can use it with the graphics operations you would usually use in C# or VB.NET.

```
pdfControl2.StartPage();
pdfControl2.Canvas.DrawString("Hello World",this.Font,new SolidBrush(Color.Black),40, 50);
pdfControl2.Canvas.DrawRectangle(new Pen(Color.Red, 1), 20,20,300,300);
pdfControl2.EndPage();
```

2.10.5.5 H) Mailmerge (COPY)

Mailmerge is using the property MergeFieldStart and the event OnGetText.

If MergeFieldStart is not "" and any text which is sent to the PDF engine starts with it the OnGetText event will be triggered to let You modify the text which is actually displayed in the PDF file.

```
private void pdfControl1_OnGetText(object Sender, string Name, System.Text.StringBuilder
Value)
{
	// Clear the text
	Value.Length = 0;
	// and set a new text
	Value.Append("WPCubed GmbH");
}
```

2.10.5.5 I) Methods defined by the DLL

```
// Methods defined in the DLL - see include file wPDF.H
typedef WPDFEnviroment (__stdcall WPDF_InitializeEx)(WPDFInfoRecord *Info); // WPDFInfoRe
typedef WPDFEnviroment (__stdcall WPDF_Initialize)(void);
typedef void (__stdcall WPDF_Finalize)(WPDFEnviroment PdfEnv);
typedef void (__stdcall WPDF_FinalizeAll)(void);
typedef void (__stdcall WPDF_SetResult)(WPDFEnviroment PdfEnv, int buffertype, char *buff

typedef int (__stdcall WPDF_BeginDoc)(WPDFEnviroment PdfEnv, char *FileName, int UseStrea
typedef void (__stdcall WPDF_EndDoc)(WPDFEnviroment PdfEnv);
typedef void (__stdcall WPDF_StartPage)(WPDFEnviroment PdfEnv);
typedef void (__stdcall WPDF_StartPageEx)(WPDFEnviroment PdfEnv, int Width, int Height, i
typedef void (__stdcall WPDF_EndPage)(WPDFEnviroment PdfEnv);
typedef void (__stdcall WPDF_StartWatermark)(WPDFEnviroment PdfEnv, char *Name);
typedef void (__stdcall WPDF_StartWatermarkEx)(WPDFEnviroment PdfEnv, char *Name, int Wid
typedef void (__stdcall WPDF_EndWatermark)(WPDFEnviroment PdfEnv);
typedef void (__stdcall WPDF_DrawWatermark)(WPDFEnviroment PdfEnv, char *Name, int Rotati

// Property and command
typedef void (__stdcall WPDF_SetSProp)(WPDFEnviroment PdfEnv, int id, char *Value);
typedef void (__stdcall WPDF_SetIProp)(WPDFEnviroment PdfEnv, int id, int Value);
typedef int (__stdcall WPDF_ExecCommand)(WPDFEnviroment PdfEnv, int id, int Value,char *b

// PDF Output Functions
typedef void (__stdcall WPDF_DrawMetafile)(WPDFEnviroment PdfEnv, unsigned int meta, int
typedef int (__stdcall WPDF_DrawDIB)(WPDFEnviroment PdfEnv, int x, int y, int w, int h, v
typedef int (__stdcall WPDF_DrawBMP)(WPDFEnviroment PdfEnv, int x, int y, int w, int h, H
typedef int (__stdcall WPDF_DrawJPEG)(WPDFEnviroment PdfEnv, int x, int y, int w, int h,
typedef int (__stdcall WPDF_DrawBitmapFile)(WPDFEnviroment PdfEnv, int x, int y, int w, i
```

```
typedef int (__stdcall WPDF_DrawBitmapClone)(WPDFEnviroment PdfEnv, int x, int y, int w,

// HDC Output function
typedef HDC (__stdcall WPDF_DC)(WPDFEnviroment PdfEnv); //** Get the DC of the PDF Canvas
typedef void (__stdcall WPDF_TextOut)(WPDFEnviroment PdfEnv, int x, int y, char * Text);
typedef char* (__stdcall WPDF_TextRect)(WPDFEnviroment PdfEnv, int x, int y, int w, int h

typedef void (__stdcall WPDF_MoveTo)(WPDFEnviroment PdfEnv, int x, int y);
typedef void (__stdcall WPDF_LineTo)(WPDFEnviroment PdfEnv, int x, int y);
typedef void (__stdcall WPDF_Rectangle)(WPDFEnviroment PdfEnv, int x, int y,int w,int h);

typedef void (__stdcall WPDF_Hyperlink)(WPDFEnviroment PdfEnv, int x, int y,int w,int h,ch
typedef void (__stdcall WPDF_Bookmark)(WPDFEnviroment PdfEnv, int x, int y,char *Name);
typedef void (__stdcall WPDF_Outline)(WPDFEnviroment PdfEnv, int level,  int x, int y,cha

// Attribute funtions
typedef void (__stdcall WPDF_SetTextDefaultAttr)(WPDFEnviroment PdfEnv, char *FontName, in
typedef void (__stdcall WPDF_SetTextAttr)(WPDFEnviroment PdfEnv, char *FontName, int Size
typedef void (__stdcall WPDF_SetTextAttrEx)(WPDFEnviroment PdfEnv, char *FontName, int Cha
         int Size, int Bold, int Italic,int Underline, unsigned int Color);
typedef void (__stdcall WPDF_SetPenAttr)(WPDFEnviroment PdfEnv, int Style, int Width, int
typedef void (__stdcall WPDF_SetBrushAttr)(WPDFEnviroment PdfEnv, int Style, int Color);

// Set License Key. This function only exists in the registered DLL, notz in the demo!
typedef void (__stdcall WPDF_SetLicenseKey)(unsigned long number, char *Name, char *Code)
```

2.10.5.6 Events

PDFControl publishes this events:

⊟ Generic	
OnNamedEvent	**pdfControl2_OnNamedEvent**
⊟ Information	
OnError	**pdfControl2_OnError**
OnMessage	**pdfControl2_OnMessage**
⊟ MailMerge	
OnGetText	**pdfControl2_OnGetText**
⊟ No active page	
AfterBeginDoc	
AfterEndDoc	
AfterEndPage	
BeforeBeginDoc	
BeforeEndDoc	
BeforeStartPage	
⊟ PDF page is open	
AfterStartPage	
BeforeEndPage	**pdfControl2_BeforeEndPage**

On Error is triggered for example if a bitmap cannot be converted by the PDF Engine.

OnGetText is triggered to let you specify the contents for a certain field (see MailMerge)

The Events under 'No active page' are triggered during the PDF creation. Do not use the 'Canvas' property inside your event handler for this events.

The 2 events under 'PDF page is open' is triggered to let you draw to the current PDF page while it is open.

2.10.6 wPDFControl DLL / ActiveX

Principle of PDF Creation

To use PDFControl in your application you first have to load the DLL an initialize it.

We have create a header file which makes it easy to do this in C and also provide a C++ class which does most of the tasks automatically. If you use the interface ActiveX you need to add one instance of it to your form and use the provided functions. Don't worry about header files.

First you need to initialize the DLL using **Initialize** or InitializeEx.

After you have initialized the DLL you can execute the function **BeginDoc** to start the output. When you are done you execute **EndDoc** to finalize and close the PDF file.

While the PDF file is open you can create a new page with **StartPage**. The page has to be closed with **EndPage**. While a page is active you can use the HDC (Device handle) of the PDF engine to create the output.

You can retrieve this handle with the function **DC** and use it with regular GDI drawing routines.

There are also some functions which make it easier to change the font, and the background and pen colors. If you use these functions the DC will be changes as if you would use the windows API - but they are much easier to use and you don't have to worry about HFONT, HBRUSH and HPEN GDI resources.

The PDF Engine includes a method called **ExecCommand**. This method is used by the RTF2PDF library as an interface to the integrated RTF-engine. Different command codes are used to select the intern functions.

Example C code: (uses wpdf.h. Use the macro 'LOAD_WPDF_ENGINE(dll_handle,dll_name)' to load the DLL and initialize the function pointers)

```
int dll;
if(LOAD_WPDF_ENGINE(dll, "wpdfcontrol02.dll" )=0)
{
  pdf = wpdfInitialize();
  if( wpdfBeginDoc(pdf,"test.pdf",0))
  {
      wpdfStartPageEx(pdf,512,812,0);
      TextOut (wpdfDC(pdf), 72, 72, "Hello World",  11);
      wpdfEndPage(pdf);
      wpdfEndDoc(pdf);
  }
  FreeLibrary(dll);
}
```

Example C++Code: (uses wPDF_Class.cpp)

```
CPDFExport cexp = CPDFExport(PDFENGINE,PDF_LIC_CODE,PDF_LIC_NAME,PDF_LIC_KEY);
if(cexp->EngineWasLoaded())
 {
  cexp->Initialize();
  if(cexp->BeginDoc(PdfFileName,0))
   {
```

```
    cexp->StartPageEx(PageH,PageW,0);
    Ellipse(cexp->DC(), 10,100,300,300);
    cexp->EndPage();
  } // if(cexp.BeginDoc(PdfFileName,0))
 cexp->EndDoc();
 cexp->Finalize();
}
```

2.10.7 wPDFControl Methods and Properties

wRTF2PDF also contains the properties and methods which are defined for wPDFControl.

In new projects we recommend to not use most of those, using the interface Memo, Memo.TextCursor and PDFCreator you have much more possibilities.

As reference we list here the obsolete interface description.

2.10.7.1 Properties

2.10.7.1 A) bool AutoLaunch

Set this property to true to launch the PDF reader after the creation.

2.10.7.1 B) Graphics Canvas

Let you draw on a graphics surface. This only works if you are using StartPage/EndPage

2.10.7.1 C) int CidFonts

Change the CIDFont feature, better use PDFCreator.CidFontMode.

2.10.7.1 D) string DebugPath

You can specify a directory where RTF2PDF will create EMF files with all the data it received.

2.10.7.1 E) bool DefaultLandscape

Cannot be used.

2.10.7.1 F) ePage DefaultSize

Cannot be used.

2.10.7.1 G) eEncoding Encoding

Change encoding, usually the default will be ok (no encoding)

2.10.7.1 H) string FileName

The filename for the created PDF file. Use "memory" to create the data in memory and read it out later using ResultBuffer

2.10.7.1 I) FontManager Font

Cannot be used.

2.10.7.1 J) eFontMode FontMode

See PDFCreator.FontMode

2.10.7.1 K) string InfoAuthor, InfoKeywords etc.

You can use the method PDFCreator.SetProperty to change the PDF info.

2.10.7.1 L) eInputFileMode InputFileMode

This property is obsolete in Version 3. It cannot be used anymore. Also InputFileName is obsolete.

2.10.7.1 M) eJPEGCompressMode JPEGCompressMode

Change the integrated JPEG compression. Please note that embedded JPEG images will not be compressed - the original JPEG data will be used.

2.10.7.1 N) string MergeFieldStart

This property is only for wPDFControl, it cannot be used in RTF2PDF.

2.10.7.1 O) PDFEncryption / Permission

in the OCX this is property "SECURITY_PERMISSION"
This property can be used to switch off copying or printing for the created PDF file. If you don't specify an owner password it will be auto-created once you set this property<>0.

Bit 1: enabled,
bit 2: Enable Printing,
bit 3: Enable Changing,
bit 4: Enable Copying,
bit 5: Enable Forms.
bit 6: LowQualityPrintOnly
bit 7: EnableDocAssembly
bit 8: EnableFormFieldFillIn
bit 9: EnableAccessibilityOp

You can also use SetIProp(pdf, 5, x);

Don't forget to also set bit 1 in property "Permission" to make the encryption work.

If you use the new PDFCreator interface, please see "Protection"

2.10.7.1 P) string OwnerPassword

The OCX uses the property SECURITY_OWNERPASSWORD

This is the owner passwords for the created PDF file. It is required to edit the PDF file in acrobat.

Don't forget to also set bit 1 in property "Permission" to make the encryption work.

If you use the new PDFCreator interface, please see PDFCreator.OwnerPW

2.10.7.1 Q) string UserPassword

The OCX uses the property SECURITY_USERPASSWORD

This is the "user" password for the created PDF file. It will be required to open and read the PDF file.

Don't forget to also set bit 1 in property "Permission" to make the encryption work.

If Permission=1 and UserPassword is empty the file will be encrypted with the PDF default password.

If you use the new PDFCreator interface, please see PDFCreator.UserWP.

2.10.7.1 R) ePageMode PageMode

The page mode to be used when the PDF file is opened.

2.10.7.1 S) int PDFAMode

Enable PDF/A - See PDFCreator.PDFAMode

2.10.7.1 T) eDevMode PDFMode

Change the internal PDF engine.

2.10.7.1 U) ePDFOptions PDFOptions

Change the internal PDF engine.

2.10.7.1 V) ePDFSecurity PDFSecurity

If you use the new PDFCreator interface, please see PDFCreator.Security.

2.10.7.2 Methods

Closes the PDF engine. Dispose should be called!

```
protected override void Dispose(bool disposing);
```

Open and close a new PDF file:

```
public bool BeginDoc();
public bool BeginDoc(Stream OutStream);
public bool BeginDoc(string FileName);
public bool BeginDocMem();
public void EndDoc();
```

Create an outline item - you can mark a line in the text to be an outline.

```
public void Outline(string Text);
public void Outline(Rectangle Rect, string Text);
public void Outline(string Text, string Bookmark);
public void Outline(int x, int y, string Text);
public void OutlineChild();
public void OutlineParent();
public void OutlineParent(int levels);
```

Create a bookmark - such an bookmark can be a jump destination for a hyperlink. Bookmarks in the text will be used, too.

```
public void Bookmark(Rectangle Rect, string BookMark);
public void Bookmark(int x, int y, string BookMark);
```

Export an image (any image in the text will be exported. See InputImage)

```
public void DrawBitmapClone(int x, int y, int w, int h, int GraphicId);
public int DrawImage(Image image);
public int DrawImage(string FileName);
public int DrawImage(Image image, int ZoomValue);
public int DrawImage(string FileName, int ZoomValue);
public int DrawImage(Image image, int x, int y, int w, int h);
public int DrawImage(string FileName, int x, int y, int w, int h);
public void DrawMetafile(Metafile metafile);
public void DrawMetafile(Metafile metafile, int xres, int yres);
```

```
public void DrawWatermark(string Name);
public void DrawWatermark(string Name, int Rotation);
```

Embed data - better embed a data object in the text.

```
public void EmbedData(Stream DataStream, int X, int Y, int W, int H, string Params, int Options);
public void EmbedDataMem(IntPtr DataPtr, int DataLen, int X, int Y, int W, int H, string Params, int Options);
```

Create a hyperlink area - better mark the text directly.

```
public void Hyperlink(Rectangle Rect, string BookMark);
public void Hyperlink(int x, int y, int w, int h, string BookMark);
```

Start and end a page

```
public bool StartPage();
public bool StartPage(Image image, int ZoomValue);
public bool StartPage(ePage format, bool landscape);
public bool StartPage(int w, int h, bool landscape);
public void EndPage();
```

Start and end a watermark

```
public bool StartWatermark(string Name);
public bool StartWatermark(string Name, ePage format);
public bool StartWatermark(string Name, int w, int h);
public void EndWatermark();
```

Write text and graphics

```
public void TextOut(string Text);
public void TextOut(string Text, int x, int y);
public void TextOut(string Text, int x, int y, int w, int h);
```

Flush the Graphics buffer:

```
public void CloseCanvas();
```

Send command id with parameters

```
public int Command(int id);
public int Command(int id, int Value);
public int Command(int id, string Text);
public int Command(int id, int Value, string Text);
public int Command(int id, int Value, IntPtr Buffer, int BufferLen);
```

Change integer and string properties

```
public void SetSProp(int Name, string Value);
public void SetIProp(int Name, int Value);
```

2.10.8 License

License Agreement for the wPDFControl and wRTF2PDF
Version 2 and Version 3
Copyright (C) 2003-2009 by WPCubed GmbH Germany
WEB : http://www.pdfcontrol.com
mail to: support@pdfcontrol.com

***** General**

The software supplied may be used by one person on as many computer systems as that person uses. Group programming projects making use of this software must purchase a copy of the software for each member of
the group.

Contact WPCubed GmbH for "Team" and "Site" licensing agreements. This documentation and the library are provided "as is" without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and/or suitability for a particular purpose. The user assumes the entire risk of any damage caused by this software.

In no event shall Julian Ziersch or WPCubed GmbH be liable for damage of any kind, loss of data, loss of profits, interruption of business or other pecuniary losses arising directly or indirectly from the use
of the program.

Any liability of the seller will be exclusively limited to replacement of the product or refund of purchase price.

***** Demo Version**

The demo version of the library may only be used to evaluate this product. The projects which were created using the demo version may not be distributed. The demo version will show a nag screen - this makes also the DLL much larger. The demo also includes the RTF engine RTF2PDF - which is not included in the standard wPDFControl.DLL!

The demo displays a nag screen and prints a watermark.

***** wPDFControl Standard License**

This License enables you to use our PDF engine technology in all your products and distribute it to your customers without the need to pay any royalties. Important: You may not distribute any of the included source files or object files or use the technology in a module (ActiveX, COM, VCL ...) which can be used by other developers in any
kind of programming language or developing environment or which can be embedded into other programs. This also prohibits the use our technology in universal PDF creation tools like a virtual printer driver or a universal RTF to PDF converter which is used from the command line.

The DLL may only be used to create PDF files from the data processed by the same application or application environment. Unless you have purchased the server license you may not use the wPDF technology in any programs (services, CGIs, ActiveServers ...) which will work on Internet servers to create PDF files to be distributed
over the WEB. As "Creator" of the PDF file always "wPDF by WPCubed GmbH" will be written to the created PDF file.

Please note that you have to send you license key to the DLL to activate it. Please use wpdfSetLicenseKey()

***** wPDFControl Server License**

In Addition to the standard license you may also use the PDF creation technology in any programs (services, CGIs, ActiveServers ...) which will work on one Internet server to create PDF files to be distributed over the WEB.

3 Document and text property API reference

In this chapter we talk about the properties of documents.
What attributes are possible, such as page size, fonts, colors and how they are controlled by TextDynamic.

This chapter should make it easy for the developer to locate the methods and interfaces which are required to modify the contents of the document.

3.1 Page Size

Like all word processor TextDynamic let You set up a certain page size.

More unusual is having different page sizes within one document. TextDynamic can do this, too.
It is possible to change the page orientation within the document, and also change the margins.

It is also possible to show lines to mark the non printable area or similar, fixed margins. Such line and their positions can be controlled through Memo.TextCommand(24,..)

You can also use a special label printing feature. Here the page is subdivided into rectangles. Each "page" is printed to one of the rectangle.

3.1.1 General

The "global" page size is defined by the interface IWPPageSize.

It is accessible through property Memo.PageSize.

The property PageSize is used to modify the current document. You can use the method MakeDefault to copy the settings to the storage for the default page size. Then this page size is applied when the text is cleared to create a new document. Consequently ReadDefault is used to read the current default page size.

3.1.2 Different in one document

If you need different page sizes within one document You can
a) use the PageSizeList
b) use the OnMeasurePage event
c) use Sections

3.1.3 Sections

Sections are used to use formatting options which are only valid within a certain part of a document. For example one section may use landscape, the next may use portrait page orientation. It is also possible that a certain header or footer is used for one section only.

A document may have multiple sections. A section always starts with a paragraph, it is not possible to change sections within one paragraph.

A new section is started with TextCursor.InputSection.

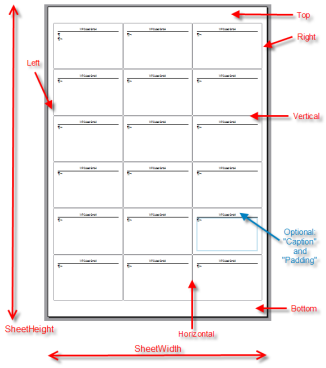
This method returns a reference to a IWPPageSize interface which can be used to modify the properties of the new section.

When you want a certain property to be valid only for this section, You need to add a bit to its "Select" property. That property can be accessed through IWPPageSize.GetProp and IWPPageSize.SetProp.

The ID of the section is also available through GetProp/SetProp. This ID can be used in Memo.BlockAdd to create a header or footer which is only used by that individual section.

3.1.4 Labels

Using the interface LabelDef you can quickly print labels. It is also possible to preview the label sheets just like they would be printed. It is even possible to edit the text on the label sheets. Also see `IWPLabelDef`.



3.2 Contents

A document can contain a lot of different elements.

First of all the text. Such text consists of characters which are combined in paragraphs.

The paragraphs may be part of a table cell, of a layer, such as a header or footer text, a footnote or the contents of a text box. (Footnotes and textboxes are only supported by the premium edition).

It is important to know, that each character may have its individual set of attributes.

The attributes are not stored in a record which belongs to the character but using a 3 bytes of a 4 byte index value to an attribute cache. This index value is referred to as "CharAttr". The 4th byte is used to store certain flags which are temporarily used for characters, most important to support spellcheck with curly underlines. The CharAttr index values can be calculated by "`AttrHelper`".

Up to 16 different attribute types are supported for individual characters. Some of this attributes are reserved.

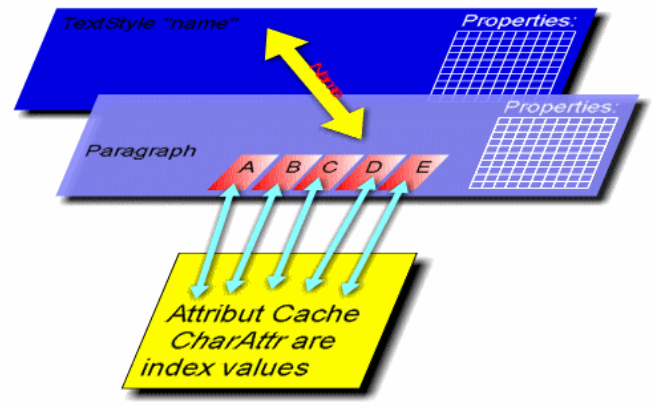
This are:
1: the Font, such as "Arial"
2: the charset, such as ansi, baltic, russian etc.
3: the size in points, such as 11pt or 8.5pt
4: the character width - this property is reserved
5: a special character effect, this is also reserved
6: a bit field which controls which of 15 flags are used (CharStyleMask). This flags are used for "bold", "italic" ...
7: a bit field to activate each of the 15 flags (CharStyleON)
8: a text color
9: a background color
10: the letter spacing
11: a reserved property. Please do not use it.
12: a reserved property. Please do not use it.
13: a special underline mode, such as dotted or double
14: the underline color
15: the text language code. This is not used yet.
16: a text style sheet

All the attributes can be undefined as well. In this case the value for the text is inherited by the definition in a preceding SPAN style element, the current paragraph or the paragraph style which is used by this paragraph.

Consequently each attribute defined in a paragraph style is overridden by the attributes defined for a character, if this character has such a definition.

Therefore the character style, such as "bold", "italic", use two bitfields for definition. One to enable the slot (6=mask), the other (on=7) to switch it on or off. The engine however does not need the enable flag if the flag is set in the "on" bit field. But if it is set in "mask" and not set in the "on", this means the style is forced to be off, even if a possible paragraph style has this style enabled.

This graphic shows how attributes are stored in a paragraph style, a paragraph and the attribute cache:



In this chapter we discuss how text and other elements are added to the document.
Text * Tables * Fields * Images * Header & Footer * Textboxes * Footnotes

3.2.1 Text

To enter text using code it is easiest to use IWPTextCursor.InputText or TextCursor.InputString.

The later allows it to specify a CharAttr index value (see Introduction) which can be calculated by "AttrHelper".

```
// We need this interfaces for text creation
IWPMemo Memo = WPDLLInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;
IWPAttrInterface AttrHelper = WPDLLInt1.AttrHelper;

// Calculate a character style index value
AttrHelper.Clear();
AttrHelper.SetFontface("Times New Roman");
AttrHelper.SetFontSize(11);
AttrHelper.IncludeStyles(2); // Italic
int mycharattr = AttrHelper.CharAttrIndex;

// and insert some text
TextCursor.InputString("Hello World", mycharattr);
```

You can also load a formatted string (RTF or HTML) at cursor position. This can be done using IWPTextCursor.InputHTML or the load methods with the "insert" flag set to true: Memo.LoadFromFile, LoadFromStream, LoadFromString, LoadFromVar.

If you want to change the text in the current paragraph it is also possible to access it directly using the Memo.CurrPar interface. This interface contains methods such as IWPParInterface.AppendText and IWPParInterface.SetText. Accessing the IWPParInterface can be very useful if you want to change or evaluate the text without moving the cursor position. To do so use the method Memo.EnumParagraphs to process all paragraphs in the text. Alternatively You can also "move" CurrPar by using its "Low level move Methods".

To move the cursor, the insertion mark, use CPPosition or the other "CP" properties, or methods such as IWPTextCursor.CPMoveBack and IWPTextCursor.CPMoveNext.

There are also methods to find text (IWPTextCursor.FindText), fields (IWPTextCursor.MoveToField), bookmarks (IWPTextCursor.MoveToBookmark) and tables (IWPTextCursor.MoveToTable). The method IWPTextCursor.MovePosition can be used to skip words or move to the start or end of a line.

To temporarily store the cursor position use the "marker API". The markers are managed in a way, that if you use InputString to insert text before a marker, the marker position will be automatically incremented. See IWPTextCursor.MarkerCollect, IWPTextCursor.MarkerDrop and IWPTextCursor.MarkerGoto.

3.2.2 Tables

Tables in TextDynamic are represented by nested paragraphs. The table, table row and cell are paragraph objects which are children of each other. cells in the same rows are siblings to each other, as are rows.

The structure is so just as it is known from HTML:

```
<table>
  <tr>
    <td>A</td><td>B</td>
  </tr>
</table>
```

TextDynamic supports 4 different ways to create a table by code. We offer so many possibilities because the data which has to be placed into the table cells can come in different ways and order. In general we favourize the use of a callback to fill the cell text, but it is not always possible to callbacks. The possibility to select from different methods makes it easy to adapt existing logic to work with TextDynamic.

Method 1 - use callback:
Call TextCursor.AddTable and use the event OnCreateNewCell to format and fill the cell. If you do not know the count of rows in advance pass 100000 and abort the creation loop inside the OnCreateNewCell event using the variable parameter "AbortAtRowEnd".

Tip: Alternatively to the callback you can use the method ASetCellProp to modify a group of cells after the complete table was created.

Advantage: Table is created automatically, only cells which need modification need 'attention'.
Disadvantage: Callback function can be difficult to read and maintain. Sometimes callback is not possible (script languages)

Method 2 - simulate user input:
Call TextCursor.AddTable - then use the properties CPTableRowNr, CPTableColNr to "move around" and insert text using InputString. You can also move the cursor using CPMoveNextRow, CPMoveNextCell which would be faster.

Advantage: Table is created automatically, only cells which need modification need 'attention'.
Disadvantage: can be slow with large tables

Method 3 - create from top to bottom:
Use TextCursor.InputTable, then InputRowStart, as many InputCell ass needed and InputRowEnd to close the current row. Create new row with InputRowStart and so on.

Advantage: fast, easy to understand logic
Disadvantage: can create rows with uneven count of columns

Method 4 - work with objects:
Create a new paragraph or modify Memo.CurrPar to make it a table object: par.SetParType((int) ParagraphType.Table). Now you can use AppendChild to create a new row and for each row use AppendChild to create a new cell. To process a different paragraph you can either use the Select methods, or you save the paragraph ID and use par.SetPtr(id).

Please see the example code in topic IWPDataBlock.AppendParagraph.

Advantage: Table can be created without change of cursor position
Disadvantage: Difficult to understand, can create rows with uneven count of columns. Exceptions are possible when SetPtr() is not correctly used.

3.2.3 Fields

TextDynamic supports two different kind of fields. First the fields which are text located with field start and -end markers. Such text can span multiple lines, paragraphs even pages. And there are fieldobjects. This fields are handles as a single character and cannot span multiple lines.

The first kind of fields are used for merge fields and edit fields. Use TextCursor.InputField to create such a field.

To create a field object use IWPTextCursor.InputFieldObject.

If you need to modify the contents of all fields in the text the obvious way to do it is using

Memo.MergeText. But You can also use EnumTextObjects to trigger an event for all markers of a certain kind in the text. Inside the event You can read and change the contents of the field.

If you need to avoid the callback event You can use CPMoveNextObject.

3.2.4 Images

TextDynamic supports BMP, EMF, JPEG and PNG images.

(The premium license provided it can also *write* multipage TIFF files - see GetPageAsBitmap.)

Images in TextDynamic can be inserted using , IWPTextCursor.InputImage and IWPTextCursor.InputPicture.

Inside the mail merge event (OnFieldGetText) it is also possible to load an image directly into a field using the "Contents" functions LoadImage and LoadPicture.

Images are handled by objects which are attached to the same textobjects which are also used to represent fields and field objects. The objects are handled differently then, i.e. they do not use properties such as "embedded text".

But You can access the image data using the IWPTxtObj interface. Usually You will use CurrObj to access the object at cursor position, or CurrSelObj to access the currently selected object.

Images in the text can be positioned as characters (default) or relative to the current paragraph or page. When an image is positioned relatively to a paragraph,it may not be positioned before its anchor.

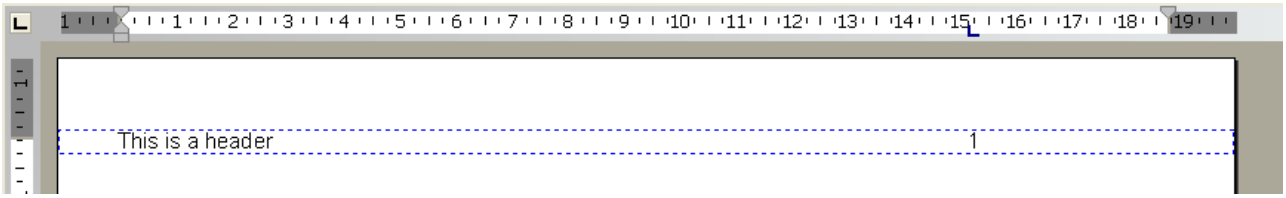
The positioning mode can be changed using the property PositionMode.

You can also activate drag&drop of images into the text (Memo.SetBProp(0,11,1)). In case it is activated you can select the position mode for the inserted images (SetBProp wpAcceptFilesOptions).

3.2.5 Header & Footer

Here we talk about the header and footer areas which may be printed on certain or all pages.

They are implemented as layers which means in general they need to be activated to be editable.
Here You see a header text in edit mode.



Internally headers and footers are implemented as "RTF Data Block" objects, just like the body of the document.
This API is used to add and find such objects: BlockAdd, BlockAppend, BlockFind
To delete a block (such a header or footer text) call the "Delete" method published by the interface IWPDataBlock.

You can also use this API to create header and footers:
IWPTextCursor.InputFooter
IWPTextCursor.InputHeader

To leave the header or footer use IWPTextCursor.GotoBody.

Normally the user can switch any header or footer element into edit mode by double clicking on it. This can also be switched off, using Memo.SetBProp(2,3,1), also see wpEditOptionsEx. While editing the body, the user sees the header and footer in shades of gray. This mode can be disabled using Memo.SetBProp(6, 19, 1), also see wpViewOptions.

To switch the layer into edit mode under program control, set the property WorkOnText to true. Now all text insertion and manipulation (InputText ...) is performed in the selected RTFDataBlock.

You can now insert page numbering:

```
IWPTextCursor TextCursor;
TextCursor = WpdllInt1.Memo.TextCursor;
TextCursor.InputFieldObject("PAGE", "", "1");
TextCursor.InputText("/");
TextCursor.InputFieldObject("NUMPAGES", "", "9");
```

There is also a dialog to create, locate and delete header and footer elements.

It is displayed by the wpaDiaManageHeaderFooter Action. You can show it with `wpaProcess("DiaManageHeaderFooter", "")`.

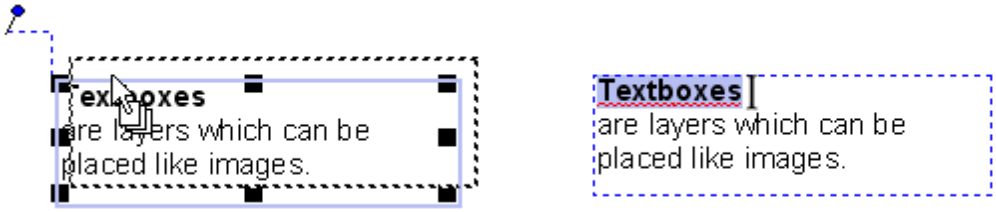
A header text may also contain images. If this images are managed relatively to the page they can also be moved down the body of the document. This feature can be used to create watermarks. (BTW: The integrated PDF engine will make sure, the image is not embedded only once.)

To create a header or footer only for a certain section, the section ID may be specified for the layer.

3.2.6 Textboxes

Text boxes are movable layers with text. You need the "premium" license to use them.

They can be displayed in two states: Selected and movable or in edit mode.

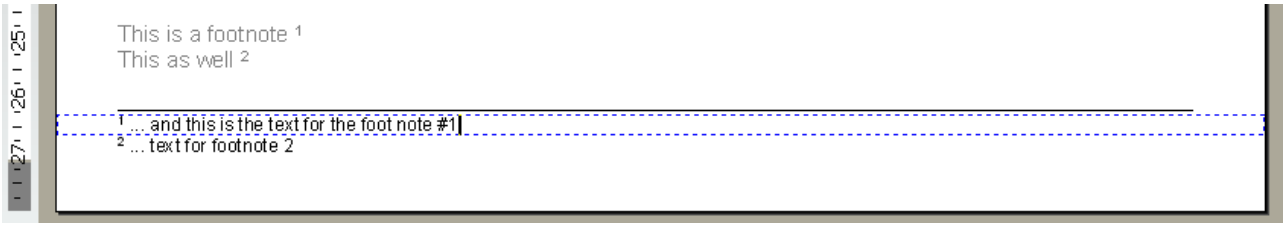


To create a text box use the API IWPTextCursor.InputTextbox. Internally they are also managed as "RTF Data Blocks".

To leave the header or footer use IWPTextCursor.GotoBody.

3.2.7 Footnotes

Footnotes are only available with the "premium" edition. They consist of a reference in the text, the footnote number and the text layer, which is automatically placed at the bottom of the page with the reference.



You can create a new footnote with TextCursor.InputFootnote. After the creation of a header, footer, textbox or footnote texts you can use GotoBody() to return to the body text to create more text there. When editing the footnote, just press ESCAPE.

3.3 Font Attributes

In this topic we want to draw Your attention to various possibilities how to change the character attributes.

i.e. we want to insert the text "this is **bold** text".

Solution A) - here we simulate typing into the editor.

(please click on the highlighted names to get more information about the respective methods and the parameters)

```
IWPMemo Memo = wpdllInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;
IWPAttrInterface Attr = Memo.TextAttr;

Attr.Clear();
Attr.SetFontface("Courier New");
Attr.SetFontSize(11);
TextCursor.InputText("this is");
Attr.IncludeStyles(1);
TextCursor.InputText("bold");
Attr.ExcludeStyles(1);
TextCursor.InputText(" ");
Attr.SetColor( wpdllInt1.ToRGB(Color.Red) );
Attr.IncludeStyles(4);
TextCursor.InputText("text");
```

BTW: To *retrieve* the current fore color use

```
int rgb =0;
if(Attr.GetColor(ref rgb))
some_control.ForeColor = ColorTranslator.FromWin32(rgb);
```

Solution B) - change attributes of existing text.

Here we assume the text "this is **bold** text" has been already written to the document. Now update the attributes of this text:

```
IWPMemo Memo = wpdllInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;
bool ok = false;

int cp = TextCursor.MarkerDrop(0);
// Use FindText.- If "MoveCursor=false", the text will be selected
```

```
TextCursor.FindText("this is bold text", true, false, false, false, ref ok);
if (ok)
{
   // We need to initialize AFTER the selection in case we use TextAttr,
   // alternatively we could use CurrSelAttr.
   IWPAttrInterface Attr = Memo.TextAttr;
   Attr.Clear();
   Attr.SetFontface("Arial");
}
TextCursor.MarkerGoto(true, cp);
```

Please see IWPTextCursor.FindText for another example.

Solution C) - use CharAttr index values and the CurrParText interface to add text quickly.

Here we first create a set of index values which are used in Append.

```
IWPMemo Memo = wpdllInt1.Memo;
IWPDataBlock Block = Memo.ActiveText;
IWPAttrInterface Attr = wpdllInt1.AttrHelper;

int header1, header2, body;
Attr.Clear();
Attr.SetFontface("Arial");
Attr.SetFontSize(11);
Attr.IncludeStyles(1);
header1 = Attr.CharAttrIndex;

Attr.SetFontSize(10);
header2 = Attr.CharAttrIndex;

Attr.ExcludeStyles(1);
body = Attr.CharAttrIndex;

for(int i=0; i<100; i++)
{
    IWPParInterface par = Block.AppendParagraph();
    if (par != null)
    {
        par.AppendText("This is Caption 1", header1);
        Block.AppendParagraph();
        par.AppendText("This is Caption 2", header2);
        Block.AppendParagraph();
        for (int j = 1; j < 30; j++) par.AppendText("some text ", body);
        wpdllInt1.ReleaseInt(par);
    }
}
wpdllInt1.ReleaseInt(Block);
Memo.ReformatAll();
```

Solution D) change the attributes of all characters in one paragraph

Here You can use
IWPAttrInterface Attr = Memo.CurrParAttr

This will change the character attributes of all the characters in one paragraph. You can add

and remove properties, the other properties will not be changed. Please note, that with this property the character attributes are changed, not the attributes stored in the paragraph or a paragraph style. To change those use CurrPar.ParASet in IWPParInterface.

3.4 Paragraph Attributes

TextDynamic supports various paragraph attributes. Not only indents and spacing, but also paragraph color, shading and numbering.

Memo.CurrPar can be used to indent the current paragraph:

```
IWPMemo Memo = wpdllInt1.Memo;
IWPParInterface Attr = Memo.CurrPar;

Attr.ParASet((int)WPAT.IndentLeft, 360);
Attr.ParASet((int)WPAT.IndentFirst, -360);
wpdllInt1.Focus();
```

It is also possible to modify **all selected paragraphs**.
To do so Attr.SetProp(1,1) must be called before the process to create a paragraph list, and SetProp(1,0) must be called to clear it.

In this example we also modify the tab stops and insert text in each paragraph.

```
IWPMemo Memo = wpdllInt1.Memo;
IWPParInterface Attr = Memo.CurrPar;

Attr.SetProp(1, 1);
try
{
    Attr.ParASet((int)WPAT.IndentLeft, 1440);
    Attr.ParASet((int)WPAT.IndentFirst, -1440);
    Attr.TabAdd(1440, 0, 0, 0);
    // Insert text at the start of each paragraph
    Attr.InsertText(0, "-\t", -1);
}
finally
{
    Attr.SetProp(1, 0);
}
Memo.Reformat();
wpdllInt1.Focus();
```

Most of the IWPParInterface methods and properties can work with a paragraph list.

We added the icon 

to the topics of the methods

which do.

When using ParASet You need to use an id (WPAT_code) to identify the attribute plus the value.

This are the most important IDs for paragraph attributes:

WPAT_IndentLeft = 17 - the left indent in twips
WPAT_IndentRight = 18; - the right indent in twips
WPAT_IndentFirst = 19; - the first indent in twips (can be negative, too)
WPAT_SpaceBefore = 20 - the space before a paragraph
WPAT_SpaceAfter = 21; - the space after a paragraph
WPAT_LineHeight = 22; - the LineHeight in in % (Has priority over WPAT_SpaceBetween)
WPAT_SpaceBetween = 23; - the space between paragraphs. When negative : Absolute, When Positive minimum

// padding, only in tables:
WPAT_PaddingLeft = 24 - Distance from Border to Text in twips (CSS = padding) tscellpaddt / trpaddl
WPAT_PaddingRight = 25 - Distance from Border to Text (CSS = padding)
WPAT_PaddingTop = 26- Distance from Border to Text (CSS = padding)
WPAT_PaddingBottom = 27- Distance from Border to Text (CSS = padding)

// Alignment
WPAT_Alignment = 29 - horizontal alignment: 0=left, 1=center, 2=right, 3=justified
WPAT_VertAlignment = 30 - vertical alignment: 0=top, 1=center, 2=bottom

This code will create bullets for the selected paragraphs

```
IWPMemo Memo = wpdllInt1.Memo;
IWPAttrInterface Attr = Memo.TextAttr;
IWPNumberStyle style;
style = Memo.GetNumberStyle(-1,1,0); // Bullet

if(style!=null)
{
    style.TextA = "¢"; // after text = "."
    style.Font="WingDings";
    style.Indent = 360;
}
Attr.AttrSET((int)WPAT.NumberSTYLE, style.ID);

wpdllInt1.ReleaseInt(style);
```

3.5 Paragraph Styles

A paragraph style includes font, color and spacing settings applied to an entire paragraph. The attributes of the style will only be used, if they were not "overridden" by either the paragraph attributes or the character attributes.

This code creates a new style:

```
IWPMemo Memo = wpdllInt1.Memo;
IWPParInterface PAttr = Memo.CurrStyle;
```

```
IWPAttrInterface CAttr = Memo.CurrStyleAttr;

Memo.SelectStyle("Heading1");
PAttr.IndentLeft = 1440;
CAttr.SetFontface("Tahoma");
CAttr.SetFontSize(12);
```

With this code the **current or the selected paragraphs** will use the new styles:

```
IWPMemo Memo = wpdllInt1.Memo;
IWPParInterface Attr = Memo.CurrPar;

Attr.SetProp(1, 1); // --> Work with selection
try
{
    Attr.StyleName = "Heading1";
    Attr.ClearCharAttr();
}
finally
{
    Attr.SetProp(1, 0);
}
Memo.Reformat();
wpdllInt1.Focus();
```

Please note that the attributes in a style are only used, if they are not overridden by the paragraph or character attributes. Here we use ClearCharAttr to make sure, the characters do not have their own attributes.

For a more selective control apply the style using ParStrCommand(9, x, name). If bit 1 is set in X, the character attributes, which are used by the selected style, are cleared, if bit 2 is set, the paragraph attributes are cleared.

So, we should better use this code:

```
IWPMemo Memo = wpdllInt1.Memo;
IWPParInterface Attr = Memo.CurrPar;

Attr.SetProp(1, 1); // work with selected text
try
{
    // Select the style and remove the attributes which would
    // otherwise override the setting in the style sheet
    Attr.ParStrCommand(9, 3, "Heading1");
}
finally
{
    Attr.SetProp(1, 0);
}
Memo.Reformat();
wpdllInt1.Focus();
```

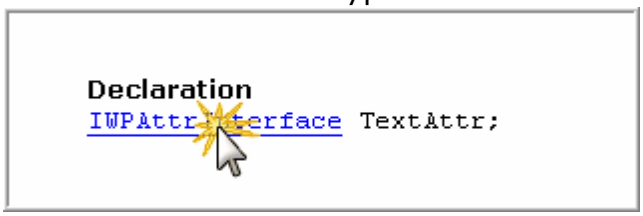
4 Interfaces (Technical API Reference)

This chapter contains detailed description of the interfaces used by TextDynamic. The object RTF2PDF is located in the assembly wPDF.dll. You need the RTF2PDF license to be able to use, the object will not work with wPDFControl.

The interfaces IWPMemo and IWPTextCursor are probably the most important.
The first includes methods to load and save, the second is used to insert text and images and control the cursor .

Tip:
As introduction to the API please read "Getting Started" first.
We also included the "Document and text property reference". Here You can look up properties and interfaces to change certain attributes usualy to documents.

There are also lots of hyperlinks in this manual:



Click to check the interface properties published by "TextAttr".

The concept of TextDynamic and RTF2PDF/TextDynamic Server has been created to make it easy to add additional functionality without having to modify the basic interfaces which could break existing applications.

For this we added a number of "Command" methods which offer additional functionality if we come to the conclusion there is a need.

So please check out this methods:

Command and CommandEx - modify the GUI, Spellcheck and Security

Memo.TextCommand - utility methods. Also tabstop and pagesize + frame line methods.

Memo.TextCommandStr - includes various important methods.

Report.Command - for additional features to "CreateReport"

4.1 General: WPDllInt + IWPMemo/IWPEditor

The WPDLLInt object is the central part of the TextDynamic word processing component. It publishes some methods to initialize the control and several interfaces. Most important are the interfaces Memo and Memo2.

To load the the DLL use SetDLLName
To initialize the engine and set Your License keys use EditorStart

To initialize the editor and select the editing mode (single editor, split screen...) use SetEditorMode.

The interface Memo is used to access and program the editor #1.

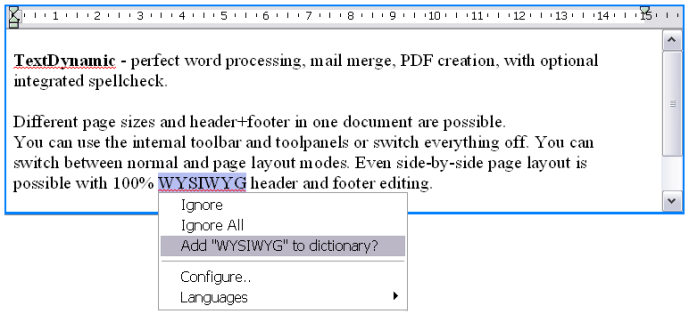
The TextDynamic control also includes a secondary editing engine. It can be use to

display thumbnails, display the result of a reporting or mail merge process or it can be used to display the same text as Editor #1 at a different location and zoom setting ("splitscreen").
The second engine can be programmed through interface Memo2.

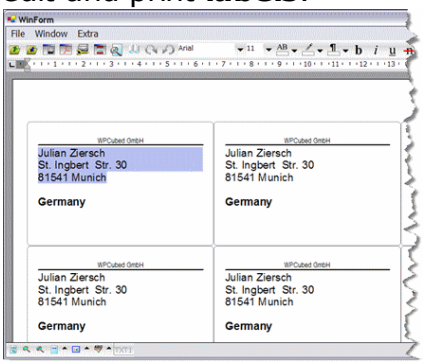
Since in certain modes the user can work freely with both editors, You can use interface CurrMemo to program the editor which is currently active. The objects Memo, Memo2 and CurrMemo all include the properties and methods of the in case of IWPMemo interface.

The interface IWPMemo publishes sub interfaces to change the font and paragraph attributes of the text (TextAttr, CurrAttr, CurSelAttr), to move the cursor and insert text or data (TextCursor). It includes methods to load and save the text (LoadFromFile ...), methods modify the behavior and display of the editor (SetBProp, SetIProp) and properties to change the layout and display. (LayoutMode, WordWrap, AutoZoom).

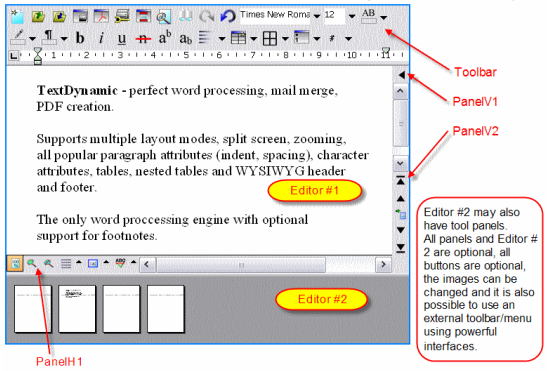
Use TextDynamic to edit database memo fields,



you can use Memo.LabelDef to quickly create, edit and print **labels**.



or for full featured word processing:



Work with the powerful mail merge, create documents under program control using the TextCursor API or the optional reporting.

Quick Links:
- Configure the Editor
- Getting Started
- You can access Memo.PageSizes to format the text to let it fit into a series of

rectangles. Each page can be retrieved easily as a metafile.
- To convert the text to PDF using the integrated PDF converter
- To prepare and send e-mails use the property Mapi.
- In one package we deliver an OCX to be used in Visual Basic 6, MS Access or Visual FoxPro (or other development tools with a decent Active X support) and a .NET assembly (written in C#) to be used in dotNET applications (C#, VB.NET, Delphi. NET). The source for the .NET assembly is provided with the license.
- For Syntax Highlighting see Memo.TextCommandStr
- Also see "Categories", "Tasks", "Mailmerge" and "Reporting"

TextDynamic Product Pages .NET / OCX

In both cases the word processing is performed by an optimized native windows DLL, the TextDynamic kernel. When using the OCX please specify the path to the DLL in property DLLName.

When using .NET you can please use the static method SetDLLName(string) as early as possible in the program (before InitializeComponent):

```
WPDynamic.WPDLLInt.SetDLLName("S:\\Appname\\WPTextDLL01.dll");
```

This loads and fixes the engine DLL in memory. It will be used for all the editors created by the application and unloaded only when the application closes or UnloadDLL was explicitly called.

We tried to make the OCX and the .NET control work as similar as possible. But we also added some exclusive utility classes and methods to the .NET library which provide a very tight integration. So the control works nicely with .NET framework Streams and preserves the PNG data loaded into .NET Pictures objects.

The interfaces (IWPMemo, IWPTextCursor, ...) are the same in the .NET assembly and in the OCX. This makes it possible to make the source code exchangeable. In this reference we mainly include examples developed in C#.

Usually you will access the interfaces like this:

```
IWPMemo Memo = wpdllint1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;

TextCursor.InputText("Hello World");
Memo.Reformat();

wpdllint1.ReleaseInt(TextCursor); // Please use ReleaseInt to avoid problems cau
wpdllint1.ReleaseInt(Memo);
```

The events published by this control however are implemented differently. While the OCX uses an event interface the .NET wrapper of course uses delegates. The parameter list of this delegates has been slightly modified to make the integration perfect.

The PDF manual includes an introduction which shows how to create a "first" application in C#, VB.NET and VB6. It also shows how to use the important method

SetLayout to load the file which contains the description of the user interface (in TextDynamic everything can be configured, the order of the buttons, captions and images).
The method SetEditorMode is used to switch the different tool panels on and off, it is also used to activate spell check, PDF export and other optional features. (The options must be included in license).

When using the registered version you need to set your license key using the method EditorStart.

The following interfaces are supported by TextDynamic:

Name	Published by	Notes
IWPMemo	Memo Memo2 CurrMemo	Main editing interface
IWPTextCursor	TextCursor IWPMemo.TextCursor	Create text by code, move cursor
IWPPdfCreator	PdfCreator	create PDF files
IWPSpell	IWPMemo.SpellCtrl	Modify the setup of the optional spellcheck engine.
IWPMAPI	IWPMemo.MAPI	reserved to create and and send e-mails.
IWPPageSize	PageSize IWPMemo.PageSize	Change document page size
IWPTextObj	CurrObj CurrSelObj IWPMemo.CurrObj IWPMemo.CurrSelObj Contents. EmbeddedObject Contents.FieldObject CurrPar.CharObj Events: OnFieldEnter OnFieldLeave OnHyperlink OnTextObjectMouse OnLoadExtImage OnBeforeSaveImage OnAfterSaveImage OnTextObjectGetText OnEnumTextObj	Modify images and other objects, such as hyperlink and mail-merge tags.
IWPParInterface	CurrPar IWPMemo.CurrPar	Modify current paragraph. Change text and set paragraph properties.
–	CurrStyle IWPMemo.CurrStyle	Modify current style to change paragraph properties.
IWPAttrInterface	IWPMemo.TextAttr	Modify the current writing mode or change the selected text.
–	CurrParAttr IWPMemo.CurrParAttr Event OnEnumParOrStyle Event OnCreateNewCell	Modify font attributes of all characters in current paragraph.
–	CurrStyleAttr IWPMemo.CurrStyleAttr	Modify font attributes defined by current style.

–	IWPFieldContents.FieldAttr	Modify font attributes of mail merge field.
–	IWPParInterface.CharAttr()	Modify font attributes of certain character in paragraph.
–	IWPMemo.CurrAttr	Modify the current writing mode
–	CurrStyleAttr	Modify font attributes defined by current style.
–	IWPMemo.CurrStyleAttr	
IWPPrintParameter	IWPMemo.PrintParameter	Modify print options.
IWPDataBlock	Memo.BlockFind Memo.BlockAdd Memo.ActiveText Event OnEnumDataBlocks	Manage header and footer texts.
IWPFieldContents	Event OnFieldGetText	Mailmerge - replace fields with text or images.
IWPDllButton	Event OnButtonClick	Provide code for custom actions added to internal toolbars.
IWPCharacterAttr	Memo.SpecialTextAttr()	Modify appearance of hyperlinks and other "special" text.
IWPReport	Property Report	Create and manage a report DB-Description. Show report template editor.
IWPLabelDef	Property Memo.LabelDef	Edit, preview and print mailing labels.

4.1.1 Events

4.1.1.1 RTF2PDF / TextDynamic

4.1.1.1 A) OnAfterSaveImage

This event is triggered after an image was saved.

Declaration C#

OnAfterSaveImage(Object Sender, int Editor, IWPTextObj TextObj);

Declaration OCX
OnAfterSaveImage(ByVal Editor As Long, ByVal TextObj As WPTDynInt.IWPTextObj)

This event is triggered after an image was saved. You can use it to reset the changes in case you used the event OnBeforeSaveImage to temporarily update the properties of the object.

Category
Image Support

4.1.1.1 B) OnBeforeSaveImage

Declaration C#
OnBeforeSaveImage(Object Sender, int Editor, IWPTextWriter Writer, IWPTextObj TextObject, **ref** bool DontSave)

Declaration OCX
OnBeforeSaveImage(ByVal Editor As Long, ByVal Writer As WPTDynInt.IWPTextWriter, ByVal TextObject As WPTDynInt.IWPTextObj, DontSave As Boolean)

This event allows it to change the properties of an object before it is saved. You can for

example update the contents or the file name property.

Parameters	
Editor	This is the number of the editor which is triggering the event.
Writer	The interface IWPTextWriter let you examine the current writing mode and path.
TextObject	The interface IWPTextObj gives you access to the object properties.
DontSave	If this variable is changed to "true" the object will not be saved.

Category
Image Support

4.1.1.1 C) OnClear

Declaration C#
OnClear(**Object** Sender, **int** Editor)

Declaration OCX
OnClear(ByVal Editor As Long)

This event is triggered after the text was cleared. You can use it to set the default text attributes.

4.1.1.1 D) OnCreateNewCell

Declaration C#
OnCreateNewCellEvent(Object Sender, int ColNr, int RowNr, IWPParInterface CellText, IWPAttrInterface CellAttr, int EventParam, **ref** bool AbortAtRowEnd)

Declaration OCX
OnCreateNewCell(ByVal ColNr As Long, ByVal RowNr As Long, ByVal CellText As WPTDynInt. IWPParInterface, ByVal CellAttr As WPTDynInt.IWPAttrInterface, ByVal EventParam As Long, AbortAtRowEnd As Boolean)

This event is triggered by method AddTable only **if** the variable EventParam was passed with a value <> 0.
The event makes it easy to assign text and attributes to each new cell.

VB Example:

```
Private Sub AddTable_Click()
   WPDLLInt1.TextCursor.AddTable "Catalog", 3, 1, True, 1000, True, True
End Sub
Private Sub WPDLLInt1_OnCreateNewCell(ByVal ColNr As Long, ByVal RowNr As Long, ByVal Cel
   If RowNr > 0 Then
   CellText.AppendText "Cell " + Str(ColNr) + " in row " + Str(RowNr), 0
   Else
   CellText.ParShading = 30
   End If
End Sub
```

Cell 1 in row 1	Cell 2 in row 1	Cell 3 in row 1

C# Example:

We use this event handler for OnCreateNewCell. The event is triggered for each created cell. The parameter EventParam is the value which was passed to AddTable. Please note that the last parameter must be ***ref*** *bool AbortAtRowEnd* - "out" instead of "ref" will not work.

```
private void WPDLLInt1_OnCreateNewCell(object Sender, int ColNr, int RowNr,
       WPDynamic.IWPParInterface CellText,
       WPDynamic.IWPAttrInterface CellAttr,
       int EventParam, ref bool AbortAtRowEnd)
{
       CellText.SetText("some text",0);
}
```

The process is started with this code:

```
WPDLLInt1.TextCursor.AddTable(
      "data", // optional name for table
      3, //columns
      10, //rows
      true, // borders
      1, // EventParam (!=0 to trigger callback)
      false,// create header rows
      false // create footer rows
);
```

The value passed as parameter EventParam is provided to the event as well. If this parameter is 0, the event will not be triggered.

Note: To create a paragraph after the table created with AddTable call InputParagraph with Mode=2

Parameters	
ColNr	This is the number of the current column. Its starts with 1
RowNr	This is the current row number. It is -1 if it is the header row, -2 for the footer row.
CellText	The IWPParInterface makes it possible to modify the text in this cell.
CellAttr	The IWPAttrInterface allows it to set the character atributes in this cell.
EventParam	This is the user variable which was passed to the AddTable function. Please remember, if this value was set to 0, the event will **not** be triggered.
AbortAtRowEnd	If this variable has been set to true inside the event the AddTable function will stop at the row end. If requested a footer row will still be added. This variable is usefull if you do not know in advance how many rows should be added by AddTable. In this case pass a large number as row count and set AbortAtRowEnd to true when the last data row has been loaded.

Category
Callback Functions
Table Support

4.1.1.1 E) OnEnumDataBlocks

Declaration C#
OnEnumDataBlocks(Object Sender, IWPDataBlock DataBlock, int EventParam)

Declaration OCX
OnEnumDataBlocks(ByVal DataBlock As WPTDynInt.IWPDataBlock, ByVal EventParam As Long)

This event is triggered by the method EnumDataBlocks. It is useful to check all header and footer texts, and, with TextDynamic "Premium" all current text objects and footnotes. A reference to the DataBlock is passed as parameter. The parameter EventParam is the integere value which was provided to EnumDataBlocks().

Category
Callback Functions

4.1.1.1 F) OnEnumParOrStyle

Declaration C#
OnEnumParOrStyle(Object Sender, bool IsControlPar, int StartPos, int Count, IWPParInterface ParText, IWPAttrInterface ParAttr, int EventParam, **ref** bool Abort);

Declaration OCX
OnEnumParOrStyle(ByVal IsControlPar As Boolean, ByVal StartPos As Long, ByVal Count As Long, ByVal ParText As WPTDynInt.IWPParInterface, ByVal ParAttr As WPTDynInt.IWPAttrInterface, ByVal EventParam As Long, Abort As Boolean)

This event is used by the methods EnumParagraphs, EnumParSiblings, EnumSelParagraphs and EnumParStyles. It is also used by WordEnum.

In all cases it can be used to extract and modify the text in the editor in a very powerful manner. Since this event is used by so many procedures, please make sure you always use the parameter EventParam to avoid that the wrong code is executed at the wrong time.

The event can be also used to add **custom spell checking** to your application. In general we recommend to use the internal spellchecking engine. When you order this addon license you will also get a dictionary compiler to create custom dictionaries. But if you have to access your dictionary through an API, for example "bool CheckWord(string word)" you can use this event to do it. First you need to activate the custom spellchecking using Command(907, 1234). (The value 1234 is just an example, any value>0 will work.) Then you add an event handler like this:

```
private void wpdllInt1_OnEnumParOrStyle(object Sender, bool IsControlPar, int StartPos, i
 int EventParam, ref bool Abort)
 {
  if (EventParam==1234)
  {
   Abort = !(MyDictionary.CheckWord(
                ParText.GetSubText(StartPos,Count)));
  }
 }
```

The boolean Abort is set to true in case the word was not found in the dictionary. Note: To implement a pop up menu the event OnMouseDownWord can be used.

Category
Callback Functions

4.1.1.1 G) OnEnumTextObj

Declaration C#
OnEnumTextObj(Object Sender, TextObjTypes ObjType, string Name, string Command, IWPTextObj Obj, int EventParam, **ref** bool Abort)

Declaration OCX
OnEnumTextObj(ByVal ObjType As WPTDynInt.TxTextObjTypes, ByVal Name As String, ByVal Command As String, ByVal Obj As WPTDynInt.IWPTextObj, ByVal EventParam As Long, Abort As Boolean)

Category
Callback Functions

4.1.1.1 H) OnError

Declaration C#
OnError(Object Sender, int Group, int Nr, string Msg)

Declaration OCX
OnError(ByVal Group As Long, ByVal Nr As Long, ByVal Msg As String)

The event is triggered when an erro happens inside the TextDynamic engine. The error message is provided as string "Msg".

Parameters	
Group	**The error group:** errException=0 - internal exception errLogging=1 - logging message errLoggingX=2 - logging message at central points errLoggingAPI=3 - log passed parameters errErrFatal=4 - should never happen. Internal state is not ok errErrProblem=5 - problem with provided data errErrFormat=6 - unexpected format errParameter=7 - parameters are incorrect errNotImplement=8 - API not implemented
Nr	**The error number:** errMessage=9 - undefined error errExceptionMsg=10 - exception errParamIsNULL=11 - null reference errParamBufferUnexpected=12 - parameter buffer incorrect errParamNotExpected=13 - integer parameter not ok errSParamNotExpected=14 - string parameter not ok errIOExceptionMsg=15 - strem or file error errEditorIsNotInitialized=16 - data not initialized errNotImplemented=17 - not implemented API errCommandIDtooLarge=18 - incorrect command id errDontFindCommand=19 - cannot find command

	errFileNotFound=20 - File was not found errEntryNotFound=21 - IMG or XML entry not found errIllegalParameter=22 - invalid Parameter errUnknownPropertyId=23 - invalid Property ID errPropertyNotSelected=23 - no attribute selected errNoStyleSelected=24 - no style selected errCannotFindStyle=25 - style was not found errThereIsNoCurrentBlock=26 - cursor not set errXMLDataNotLoaded=27 - XML data not loaded errEditorNotInCorrectState=28 - need other API executed before errInterfaceNotDefined=29 - interface not defined errDialogNotImplemented=30 - dialog not implemented **This error numbers are also returned by the method Command()**

4.1.1.1 I) OnFieldGetText

This event is used by the reporting engine to read field values. It is also used by the mail merge function.

Declaration C#
void OnFieldGetText(Object Sender, int Editor, string FieldName, IWPFieldContents Contents)

Declaration OCX
OnFieldGetText(ByVal Editor As Long, ByVal FieldName As String, ByVal Contents As WPTDynInt.IWPFieldContents)

This event is use by the mail merge feature (see Memo.MergeText) and the reporting engine (see interface IWPReport).

It is used to fill fields with data during the process of mail merge or report creation. The mail merge is started with MergeText. Inside the merge process the event OnFieldGetText is triggered for all fields.

The provided interface Contents makes it easy to change the text displayed by the field.

It is also **possible to read the current text** in a field. To do so simply use SaveTo... with the property OnlySelection set to true. This works, because the text of a field is (invisibly) selected while the OnFieldGetText event gets called. Alternatively it is also possible to use Memo.TextCommandStr to delete, read and write fields.

VB6 Example:

```
private Sub WPDLLInt1_OnFieldGetText(ByVal Editor as long, _
      ByVal FieldName as string, _
      ByVal Contents as WPToolsInt.IWPFieldContents)
  if FieldName='NAME' then Contents.StringValue = 'Julian Ziersch'
End Sub
```

C# Example:

```
Random rnd = new Random();
```

```
private void wpdllInt2_OnFieldGetText(
      object Sender,
      int Editor,
      string FieldName,
      IWPFieldContents Contents)
{
    if (FieldName == "NAME")
                Contents.StringValue = "Julian Ziersch";
    else Contents.StringValue = Convert.ToString(rnd.NextDouble());
}
```

It is also possible to create or update an image which is placed inside the field. To make this as easy as possible there are two functions LoadImage and LoadPicture. Both methods will not simply replace the contents of the field with a new image but reuse an existing image container object.

Hint: In our product WPTools the event with similar functionality was named OnMailMergeGetText.

To create a hyperlink inside the field use InputHyperlink. You can also load a file with formatted text using LoadText

To create a table with data inside the field execute AddTable. The event OnCreateNewCell can be used to format and fill each new cell.

If you know that you do not need the field after the merge process you can execute DeleteField. The field markers, not the contents will be removed.

An interface to access the current field is provided as property FieldObject. You can use this interface to read other properties of the merge field.

Notes:
If you are using the Active-X and your developing system does not provide access to the interface passed as parameter, use the property EventField instead.

The event is also used for report group variables in the "add:" mode. Here only the StringValue may be modified. The property FieldObject will be null in this case!

The reporting engine will also trigger the event OnFieldGetText to evaluate fields which were used in band conditions such as ?fieldname=null and ?fieldname#null.
In this case the property IsMergefield =false. Use Contents.ExecStrCommand to modify the field state property.

4.1.1.1 J) OnGetSpecialText

Member of WPDLLInt
Declaration C#
OnGetSpecialText(Object Sender, int Editor, int PageNr, DataBlockKind Kind, **ref int SelectedID**);

Declaration OCX
OnGetSpecialText(ByVal Editor As Long, ByVal PageNr As Long, ByVal Kind As Long, SelectedID As Long)

This event is used to select a certain header or footer for certain pages. For an example see Name.

Category
Hader and Footer Support

4.1.1.1 K) OnInitializePar

Declaration C#
OnInitializePar(Object Sender, int Editor, IWPParInterface Paragraph)

Declaration OCX
OnInitializePar(ByVal Editor As Long, ByVal Paragraph As WPTDynInt.IWPParInterface)

This event allows it to change the properties of the paragraph and the text according to its contents. It is triggered before the paragraph is formatted.

This (kind of silly) example shades the paragraph if it contains the text "shade" somewhere and removes the shading if it does not:

```
Private Sub WPDLLInt1_OnInitializePar(ByVal Editor As Long, ByVal Paragraph As WPTDynInt.
  If Paragraph.HasText("shade", False) Then
    Paragraph.ParShading = 20
  Else
    Paragraph.ParShading = 0
  End If
End Sub
```

4.1.1.1 L) OnLoadExtImage

Declaration C#
OnLoadExtImage(Object Sender, int Editor, string LoadPath, string URL, IWPTextObj TextObj, **ref** bool Ok);

Declaration OCX
OnLoadExtImage(ByVal Editor As Long, ByVal LoadPath As String, ByVal URL As String, ByVal TextObj As WPTDynInt.IWPTextObj, OK As Boolean)

This event is triggered during text loading. If images are not imbedded but linked (only a name is in the file) this event can be used to load the image from a directory or data base.
When you load the image in your code set the variable OK to true - then the default code will be skipped (it tries to locate the image data in the same directory as the loaded file).
To load an image use the interface TextObj.

4.1.1.1 M) OnLoadExtString

Declaration C#
OnLoadExtString(Object Sender, int Editor, string LoadPath, string URL, object DataStream, **ref** bool Ok)

Declaration OCX
OnLoadExtString(ByVal Editor As Long, ByVal LoadPath As String, ByVal URL As String, ByVal DataStream As IWPStream, OK As Boolean)

This event is triggered by the HTML reader to load an external CSS style sheet.

4.1.1.1 N) OnLoadText

Declaration C#
void OnLoadText(Object Sender, int Editor, string Filename)

Declaration OCX
OnLoadText(ByVal Editor As Long, ByVal filename As String)

This event is triggered after a file was loaded. It can be used to update the caption of a window to display the current file name.

4.1.1.1 O) OnMeasurePage

Declaration C#
OnMeasurePage(Object Sender, int Editor, int PageNr, IWPMeasurePageParam Size)

Declaration OCX
OnMeasurePage(ByVal Editor As Long, ByVal PageNr As Long, ByVal Size As WPTDynInt. IWPMeasurePageParam)

TextDynamic supports different page sizes in one document using RTF sections. If you want to change the page size for certain pages using code it you should use this event. You can easily update the properties in the parameter interface Size. The modified size will be only used for the page with the number "PageNr".

Category
Modify the layout of the text display

4.1.1.1 P) OnNotify

Declaration C#
OnNotify(**Object** Sender, **int** Editor, **int** MsgID);

Declaration OCX
OnNotify(ByVal Editor As Long, ByVal MsgID As Long)

Currently this message IDs are supported:

MSGID_ChangedActiveText=21: The editor changed between header and body text.

MSGID_MouseWheelAtStart=23: The mouse wheel was used to scroll to the very start.

MSGID_MouseWheelAtEnd=24: The mousesue wheel was used to scroll to the very end.

MSGID_ChangeLastFileName=27: Last file name was changed.

MSGID_ChangeModified=28 The modified property was changed.

MSGID_SelectWatermark = 29; Application may choose a new watermark.
Current page can be read using Memo.TextCommand(25, 0, 0). See C# Example there.

4.1.1.1 Q) OnPaintWatermark

Declaration C#
OnPaintWatermark(
Object Sender,
int Editor,
int Mode,
Graphics Canvas,
float X, **float** Y, **float** X1, **float** Y1, **float** Xres, **float** Yres);

The .NET assembly uses different parameters than the OCX for this event type.

It passes the coordinates and the resolution as floating point variables and instead of a device handle (HDC) it passes a reference to a .NET Graphics object as drawing surface "Canvas".

Declaration OCX
OnPaintWatermark(ByVal Editor As Long, ByVal Mode As Long, ByVal DC As Long, ByVal X As Long, ByVal Y As Long, ByVal X1 As Long, ByVal Y1 As Long, ByVal Xres As Long, ByVal Yres As Long)

This event can be used to paint into the background of any page.

(If you need to select a watermark which was loaded using Command(143) please use OnNotify . Example.)

Parameters:

The page rectangle and specified by X,Y,X1,Y1. This values are based on the resolution specified by XRes ansd YRes.

The lower 3 bytes of parameter "Mode" is the current **page number**.

In the higher word this bits are used:

0x1000000 : we are currently printing
0x2000000 : we are painting inside the editor
0x4000000 : we are currently exporting to PDF

Example:

```
private void OnPaintWatermark(object Sender, int Editor, int Mode,
   System.Drawing.Graphics Canvas, float X, float Y,
   float X1, float Y1, float Xres, float Yres)
{
   int XMargin = (int)(Xres/2.54);
   int YMargin = (int)(Yres/2.54);
   Canvas.DrawRectangle(System.Drawing.Pens.Black, X+XMargin, Y+YMargin, X1-XMargin*2, Y1

   // Draw the page number as vertical text
   String drawString = "Page " + ((Mode & 0xFFFFF) + 1).ToString();

   Font drawFont = new Font("Arial", 11);
   SolidBrush drawBrush = new SolidBrush(Color.Red);

   StringFormat drawFormat = new StringFormat();
   drawFormat.FormatFlags = StringFormatFlags.DirectionVertical;
   // Draw in upper left corner
   Canvas.DrawString( drawString, drawFont, drawBrush, X, Y, drawFormat);
}
```

4.1.1.1 R) OnReadFormulaVar

Declaration C#
OnReadFormulaVar(Object Sender, string Name, IWPReportBand BandContext, int CountInGroup, out double Value);

Declaration OCX
OnReadFormulaVar(ByVal Name As String, ByVal BandContext As WPTDynInt.IWPReportBand,

ByVal CountInGroup As Long, Value As Double)

This event is use by the reporting engine (see interface IWPReport).

It is triggered to read the value of a variable which is used in a formula. Please don't mix up with group variables used by the reporting engine, those group variables contain the formulas which itself contain the variables which trigger this event. In fact any unknown name inside a formula will trigger the event OnReadFormulaVar.

The formula "TABLE.PRICE*TABLE.AMOUNT" will trigget the event twice, first to read "TABLE.PRICE" then to read "TABLE.AMOUNT".

Parameters	
Name	This is the name of the formula variable.
BandContext	This is the reference to the current band, it can be used to also check the parent group.
CountInGroup	This integer value represents the count of repetitions in this loop. group.
Value	This double variable is the value which should be used in place of this name. It is initialized with 0.

Category
Reporting

4.1.1.1 S) OnReportState

This event is used by the reporting engine (see interface IWPReport) to create and navigate in data sets.

Also required is even OnFieldGetText to read field values.

Declaration C#
OnReportState(**object** Sender, **string** Name, **int** State, WPDynamic.IWPReportBand Band, **ref bool** Abort)

Declaration OCX
OnReportState(ByVal Name As String, ByVal State As Long, _
ByVal Band As WPTDynInt.IWPReportBand, Abort As Boolean)

This event occurs during report creation. It gives you a chance to prepare a database query, to advance to the next record and to prepare the attributes of the bands.

Parameters

Name:
This is the name of the band.

State:
This parameter is used to check the current processing state of the reporting engine:

0 (REP_BeforeProcessGroup): A reporting group is about to be started/looped. **You need to set "Abort" to false, otherwise the group will not be processed**. Usually you will need to prepare a sub query and set "Abort" to true in case the query is empty. The event will also occur when the group is processed again - Count is >0 in this case.

1 (REP_PrepareText): A text band is prepared. It will be used next. Usually you have

nothing to do - but you can modify the properties of the band paragraphs.
After the band was *prepared*, all paragraphs will be copied to editor #2 in the TextDynamic control and after that the event OnFieldGetText will be fired for all merge fields in the text.

2 : not used

3 (REP_PrepareHeader): A header band is prepared.

4 : not used

5 (REP_AfterProcessGroupData): The group data has been processed completely. You can use this state to sum up totals.

6 (REP_PrepareFooter): A footer band is prepared.

7: not used

8 (REP_AfterProcessGroup): The group was processed. You can use this event to move to next record.

Example:

This simplified C# example code loops all groups in the report template 10 times.

It has been created to be used as template in Your Application.

```
        // This event is used to start, prepare and close a report group
        // Usually you would use the "Name" to select a certain database,
        // rewind it depending on State and set "Abort" to true
        // when the last record was reached when State=0.
        private void wpdllInt2_OnReportState(
                        object Sender,
                        string Name, int State,
                        IWPReportBand Band,
                        ref bool Abort)
        {
            switch(State)
            {
                case 0: //REP_BeforeProcessGroup
                {
                    // If Band.Count=1 take "Name" to open a data base query
and
                    // move to first Record.
                    // Set Abort to TRUE if the query has the state
"EOF" (EndOfFile)
                    // Here also Set total variables to 0.
                    Abort = Band.Count>10;
                    break;
                }

                case 1: //REP_PrepareText
                {
                    // We can process a text band depending on a condition
                    break;
                }

                case 3: //REP_PrepareHeader
```

```
                {
                    // We can process a header band depending on a condition
                    break;
                }

                case 5: //REP_AfterProcessGroupData
                {
                    // Take the named data set and post process the current
row.
                    // This event will only be triggerd if the group contained
                    // a data row. You can use it to calculate totals.
                    // (Band.Count will be -1 in this event)
                    break;
                }

                case 6: //REP_PrepareFooter
                {
                    // We can process a footer band depending on a condition
                    break;
                }

                case 8: //REP_AfterProcessGroup
                {
                    // This event takes place after the group data and footer
                    // have been processed. After this event any headers are
                    // created. (Band.Count will be -1 in this event)
                    break;
                }
            }
        }
```

Note:
If you are using the Active-X and your developing system does not provide access to the interface passed as parameter, use the property EventReportBand instead.

Category
Reporting

4.1.1.1 T) OnTextObjectGetText

Declaration C#
OnObjectGetText(**Object** Sender, int Editor, **int** PageNr, IWPTextObj TextObj)

Declaration OCX
OnTextObjectGetText(ByVal Editor As Long, ByVal CurrPage As Long, ByVal Obj As WPTDynInt.IWPTextObj)

This event can be used to display custom text inside of text objects. There are several standard objects, such as the page number, the page count, time and date fields. All this objects are - in contrast to merge fields - handled as one character only. This means the contents cannot contain a line wrap. But unlike merge fields, which really *contain* text, the text displayed by the text objects is retrieved before the object is painted. You can see the effect with the TIME field - here the current time is updated every time the screen is refreshed.

The contents of the object with an unknown name (not PAGE, NUMPAGES etc) is by default retrieved from the object property Params. This property is read from and written to RTF. Inside the event OnTextObjectGetText you can update the property Params to display custom information, for example the current file path, the name of the author or any text you need.

This VB code inserts a text object:

```
WPDLLInt1.TextCursor.InputFieldObject "DYNAMIC", "", ""
```

This event simply displays <undefined: ...>

```
Private Sub WPDLLInt1_OnTextObjectGetText(ByVal Editor As Long, ByVal CurrPage As Long, By
  Obj.Params = "<undefined:" + Obj.Name + ">"
End Sub
```

4.1.1.2 Exclusive to TextDynamic

4.1.1.2 A) OnBeforeOverwriteFile

Member of WPDLLInt
Declaration C#
OnBeforeOverwriteFile(Object Sender, int Editor, string filename, bool OnlySelection, **ref** bool Abort);

Declaration OCX
OnBeforeOverwriteFile(ByVal Editor As Long, ByVal filename As String, ByVal OnlySelection As Boolean, Abort As Boolean)

This event can be used rename an existing file (or abort the process completely) in case a file which is about to be saved, already exists.

Category
Load and Save

4.1.1.2 B) OnButtonClick

Member of WPDLLInt
Declaration C#
OnButtonClick(Object Sender, int Editor, IWPDllButton Def)

Declaration OCX
OnButtonClick(ByVal Editor As Long, ByVal Def As WPTDynInt.IWPDllButton)

This event is triggered when the user clicks on a button or menu. (new: works with all buttons, not only custom buttons).

Inside the event OnButtonClick event the property Disabled can be set to true to abort the execution of the default button action until the button is clicked the next time.

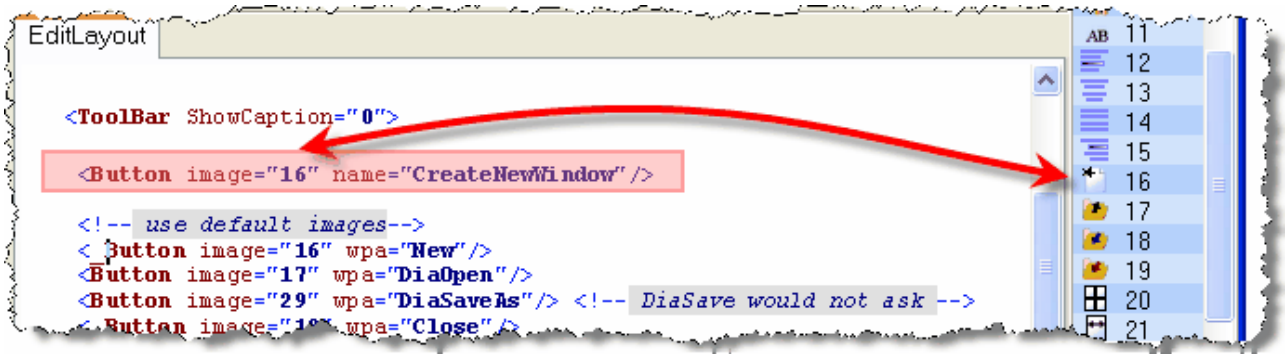
(The state of "disabled" cannot be permanently changed in OnButtonClick - this has to be done in OnUpdateGUI)

The interface IWPDllButton provides access to the properties of the button. Most important is property "Name" which can be set in the toolbar description.

Example VB6:

```
Private Sub WPDLLInt1_OnButtonClick(ByVal Editor As Long, ByVal Def As WPTDynInt.IWPDllBu
   If Def.Name = "CreateNewWindow" Then
     LoadNewDoc ' implemented as public routine in 'Main.BAS'
   End If
End Sub
```

This code uses a new button created in the PCC file:



Note:
If you are using the Active-X and your developing system does not provide access to the interface passed as parameter, use instead either the property EventButton or Memo.TextCommandStr(33,...)

4.1.1.2 C) OnChangePosition

Member of WPDLLInt
Declaration C#
OnChangePosition(**Object** Sender, **int** Editor)

Declaration OCX
OnChangePosition(ByVal Editor As Long)

This event is triggered when the position of the insertion marker (=cursor) is changed. You can use it to update the statusbar.
The position of the cursor, paragraph number, page number and similar, can be retrieved using the CP* properties published by TextCursor and CurrMemo.TextCursor.

However, in general we recommend to use OnUpdateGUI to update all status items, not only the status bar but also the enabled/disabled/checked state of menu items and buttons.

Category
Display Status Information

4.1.1.2 D) OnChangeSelection

Member of WPDLLInt
Declaration C#
OnChangeSelection(**Object** Sender, **int** Editor)

Declaration OCX
OnChangeSelection(ByVal Editor As Long)

This event is triggered when the selection has been changed. The event is triggered asynchronly after the change has been performed, the next time the application is idle.

Category
Display Status Information

4.1.1.2 E) OnChangeText

Member of WPDLLInt
Declaration C#
TextChanged(EventArgs e)
Here the standard event is used. You can cast e to wpEventArgs which contains the property Editor.

Declaration OCX
OnChangeText(ByVal Editor As Long)

The event will be triggered after the text was modified. OnChangingText will be triggered before the change.

4.1.1.2 F) OnChangeViewMode

Member of WPDLLInt
Declaration C#
OnChangeViewMode(Object Sender, int Editor, int AutoZoom, int Zooming)

Declaration OCX
OnChangeViewMode(ByVal Editor As Long, ByVal AutoZoom As Long, ByVal Zooming As Long)

This event is triggered when the view mode has been changed. This happens when the zooming or layout mode is updated.

Parameters	
Editor	This is the number of the editor which is triggering the event.
AutoZoom	0: AutoZoom Off 1: Show full page width 2: Show full page (width + height) 3: Adjust column count - display as many pages in a row as possible 4: Adjust column count and distance - display as many pages in a row as possible and adjust the distance if the pages have different sizes

Category
Display Status Information

4.1.1.2 G) OnChangingText

This event is triggered before the text is modified.

Member of WPDLLInt
Declaration C#
OnChangingTextEvent(Object Sender, int Editor, **ref** bool AllowChange)

Declaration OCX
OnChangingText(ByVal Editor As Long, AllowChange As Boolean)

When using VB6 better use the standard event **Validate(Cancel As Boolean)** which allows it to cancel the operation which would modify the text.

4.1.1.2 H) OnClick

Member of WPDLLInt
Declaration C#
Click(EventArgs e)
Here the standard event is used. You can cast e to wpEventArgs which contains the property Editor.

Declaration OCX
OnClick(ByVal Editor As Long, Handled As Boolean)

This event is triggered when the use clicks inside the editor. You can set the variable "Handled" to true in case the default code should not be executed.

4.1.1.2 I) OnClickCreateHeaderFooter

Member of WPDLLInt
Declaration C#
OnClickCreateHeaderFooter(Object Sender, int Editor, int Kind, **ref** int Range);

Declaration OCX
OnClickCreateHeaderFooter(ByVal Editor As Long, ByVal Kind As Long, Range As Long)

This event is triggered when the user clicks double into the top or buttom margin areas. You can use the variables Kind and Range if you want to create a new header or footer.
VB Example:

```
Private Sub WPDLLInt1_OnClickCreateHeaderFooter(ByVal Editor As Long, ByVal Kind As Long,
   If Kind = 1 Then
      WPDLLInt1.TextCursor.InputHeader Range, "", ""
   Else
      WPDLLInt1.TextCursor.InputFooter Range, "", ""
   End If
End Sub
```

Parameters	
Kind	This variable is 1 for a header, 2 for a footer text.
Range	This variable contains the "suggested" range : all pages, odd, even, first page.

Category
Hader and Footer Support

4.1.1.2 J) OnClickPage

Member of WPDLLInt
Declaration C#
OnClickPage(Object Sender, int Editor, int PageNr);

Declaration OCX
OnClickPage(ByVal Editor As Long, ByVal PageNr As Long)

This events is triggered when the user clicks on a page.

4.1.1.2 K) OnDblClick

Member of WPDLLInt
Declaration C#
DblClick(EventArgs e)
Here the standard event is used. You can cast e to wpEventArgs which contains the property Editor.

Declaration OCX
OnDblClick(ByVal Editor As Long, Handled As Boolean)

This event procedure is called after the user clicked twice. You can set the variable Handled to true to skip the default code.

4.1.1.2 L) OnFieldEnter

Member of WPDLLInt
Declaration C#
OnEnterField(**Object** Sender, **int** Editor, **string** Fieldname, IWPTextObj Field)

Declaration OCX
OnFieldEnter(ByVal Editor As Long, ByVal FieldName As String, ByVal Field As WPTDynInt.
IWPTextObj)

When in formular editing mode (see SetEnableFlags) this event will be triggered when the cursor was moved into a different field.

Normally this event is not used, when the editor is not in the formular editing mode. But when using Memo.SetBProp(17,6,1) the event will be also used during regular editing.

4.1.1.2 M) OnFieldLeave

Member of WPDLLInt
Declaration C#
OnLeaveField(**Object** Sender, **int** Editor, **string** Fieldname, IWPTextObj Field, **ref bool** Abort)

Declaration OCX
OnFieldLeave(ByVal Editor As Long, ByVal FieldName As String, ByVal Field As WPTDynInt.
IWPTextObj, Abort As Boolean)

When in formular mode (see SetEnableFlags) this event will be triggered before the cursor is moved into a different field. You can check the contents of the field (Field.EmbeddedText) and cancel the operation by setting varaiable Abort to true.

Normally this event is not used, when the editor is not in the formular editing mode. But when using Memo.SetBProp(17,6,1) the event will be also used during regular editing. In this case the event will be even be triggered when a field has been deleted. (Abort is without function in this case)

4.1.1.2 N) OnHyperlink

Member of WPDLLInt
Declaration C#
OnHyperlink(Object Sender, int Editor, string URL, IWPTextObj TextObj);

Declaration OCX
OnHyperlink(ByVal Editor As Long, ByVal URL As String, ByVal TextObj As WPTDynInt.

IWPTextObj)

This event is triggered when the user clicks on a hyperlink. You can use this event to open the mail application, the internet browser or select a different record in the database. Using the parameter TextObj you can also examine and modify the hyperlink: TextObj.EmbeddedText is the visible text, TextObj.Command is the URL.

Category
Hyperlinks and Bookmarks

4.1.1.2 O) OnMouseDown

Member of WPDLLInt
Declaration C#
MouseDown(MouseEventArgs e)
The .NET assembly uses the standard event type MouseDown. The passed MouseEventArgs can be casted to the wpMouseEventArgs type to read the property "Editor".

Declaration OCX
OnMouseDown(ByVal Editor As Long, ByVal Button As Long, ByVal Shift As Long, ByVal X As Long, ByVal Y As Long)

This event occurs when the user presses the mouse button. The parameter Shift can be used to check the state of the control keys. The parameter Button will be 0 for left button, 1 for right and 2 for the middle button.

Parameters	
Editor	The editor number, 1 or 2
Button	0 = left button, 1 = right, 2 = middle mouse button
Shift	A bit field representing the state of the control keys: 1 : Shift key 2 : ALT key 4 : Ctrl key
X	X position relatively to the editors upper left corner.
Editor	Y position relatively to the editors upper left corner.

4.1.1.2 P) OnMouseDownWord

Member of WPDLLInt
Declaration C#
OnMouseDownWord(Object Sender, int Editor, int Button, int Shift, int X, int Y, IWPParInterface Paragraph, int PosInPar, int Count)

Declaration OCX
OnMouseDownWord(ByVal Editor As Long, ByVal Button As Long, ByVal Shift As Long, ByVal X As Long, ByVal Y As Long, ByVal Paragraph As WPTDynInt.IWPParInterface, ByVal PosInPar As Long, ByVal Count As Long)

This event is triggered when the user clicks on any of the words in the text. It will be also fired when the context menu key is being pressed.
Usually you will use the event to display a popup menu. Using the parameter Paragraph you can read and change the text at the position the click was performed.

This event is also used by custom spellchecking which can be activated using Command(907).

If you use this event to show a custom popup dialog please use Memo.SetBProp(0, 19, -1) to deactivate the default popup dialog.

Tip: If custom spellchecking was activated using Command(907) and the word was marked to be wrong, bit 10 (value 512) will be set in the bit field "Shift". The current word will be selected in this case. In the event handler you can show a popup menu and either remove the misspell-marker or also replace the text. The misspell-marker will be removed if either Paragraph. ReplaceText or Paragraph.ReplaceCharAttr was used inside the event handler.

Note: The .NET ContextMenu Show() method already returns before the Click events of the items have been triggered, to solve this problem you need to call method Application.DoEvents after Show.

This example shows how to display a popup menu to change the current word:

```
private bool TextWasOK;
private string NewText;
private void wpdllInt1_OnMouseDownWord(object Sender, int Editor, int Button, int Shift, 
{
    contextMenu1.MenuItems.Clear();
    // Add first menu - add this word
    MenuItem men = new MenuItem();
    men.Text = "Add: " + Paragraph.GetSubText(PosInPar,Count);
    contextMenu1.MenuItems.Add(0, men);
    men.Click  += new System.EventHandler(this.SpellMen_IgnoreThisWord);
    // Add second menu - replace with this text
    MenuItem men2 = new MenuItem();
    men2.Text = "Replacement text";
    contextMenu1.MenuItems.Add(0, men2);
    men2.Click  += new System.EventHandler(this.SpellMen_ChangeWord);
    // Initialize global variables
    TextWasOK = false;
    NewText = "";
    // Display popup and wait
     contextMenu1.Show(wpdllInt1,new Point(X,Y));
    // Popup was closed
    // Now trigger the Click events
    Application.DoEvents();
    // and change the text
    if(NewText!="")
     Paragraph.ReplaceText(PosInPar,Count,NewText);
    // or just force the removal of the misspell-marker
    else if (TextWasOK)
     Paragraph.ReplaceCharAttr(PosInPar,0,0);
}

private void SpellMen_IgnoreWord(object sender, System.EventArgs e)
{
 TextWasOK = true;
}

private void SpellMen_ChangeWord(object sender, System.EventArgs e)
{
 TextWasOK = true;
 NewText = ((MenuItem)sender).Text;
}
```

4.1.1.2 Q) OnMouseMove

Member of WPDLLInt
Declaration C#
MouseMove(MouseEventArgs e)
The .NET assembly uses the standard event type MouseMove. The passed MouseEventArgs can be casted to the wpMouseEventArgs type to read the property "Editor".

Declaration OCX
OnMouseMove(ByVal Editor As Long, ByVal Shift As Long, ByVal X As Long, ByVal Y As Long)

This event occurs when the user moves the mouse over the editor.

Parameters	
Editor	The editor number, 1 or 2
Shift	A bit field representing the state of the control keys: 1 : Shift key 2 : ALT key 4 : Ctrl key Please use a logical "and" operation to check the bits!
X	X position relatively to the editors upper left corner.
Editor	Y position relatively to the editors upper left corner.

4.1.1.2 R) OnMouseUp

Member of WPDLLInt
Declaration C#
MouseUp(MouseEventArgs e)

The .NET assembly uses the standard event type MouseUp.
The passed MouseEventArgs **can be casted to the wpMouseEventArgs** type to read the property "Editor".

Declaration OCX
OnMouseUp(ByVal Editor As Long, ByVal Button As Long, ByVal Shift As Long, ByVal X As Long, ByVal Y As Long)

This event occurs when the user releases the mouse button. The parameter Shift can be used to check the state of the control keys. The parameter Button will be 0 for left button, 1 for right and 2 for the middle button.

Parameters	
Editor	The editor number, 1 or 2
Button	0 = left button, 1 = right, 2 = middle mouse button
Shift	A bit field representing the state of the control keys: 1 : Shift key 2 : ALT key 4 : Ctrl key
X	X position relatively to the editors upper left corner.
Editor	Y position relatively to the editors upper left corner.

VB6 Example:

VB6: This code checks for a click into a merge field and then updates its contents:

```
Private Sub WPDLLInt1_OnMouseUp(ByVal Editor As Long, ByVal Button As Long, ByVal Shift As
    Dim Memo As IWPMemo
    Dim Obj As IWPTextObj
    Set Memo = WPDLLInt1.Memo
    Set Obj = Memo.GetObjAtXY(X, Y, 0, -1)
    If Not Obj Is Nothing Then
       Obj.EmbeddedText = "new text"
       Memo.ReformatAll False, True
    End If
End Sub
```

C#: This event handler also checks for e merge field and opens a dialog to edit the embedded text.

```
private void wpdllInt1_MouseUp(object sender, MouseEventArgs e)
{
            wpMouseEventArgs me = (wpMouseEventArgs)e;
            IWPMemo Memo;
            if (me.Editor == 2)
                Memo = wpdllInt1.Memo2;
            else Memo = wpdllInt1.Memo;
            IWPTextCursor Cursor = Memo.TextCursor;

            //This is a simple form with a lable and a textbox + a OK and cancel Button
            Form2 f = new Form2();
            try
            {
                // Set Dialog Position
                int x = 0, y = 0;
                Memo.GetXY(8, ref x, ref y);
                f.StartPosition = FormStartPosition.Manual;
                f.Left = me.X + x;
                f.Top = me.Y + y;
                // Which Button was pressed?
                if (me.Button == MouseButtons.Left)
                    f.Text = "Left";
                else if (me.Button == MouseButtons.Middle)
                    f.Text = "Middle";
                if (me.Button == MouseButtons.Right)
                    f.Text = "Right";
                // Possibility: get Position
                // int cppos=0;
                // Memo.GetPosAtXY(2, me.X + x, me.Y + y, out cppos);

                // Find Field at current mouse position and open a form to edit
the field contents
                IWPTextObj obj = Memo.GetObjAtXY(0,0, 1, -1);

                if (obj != null)
                {
                    f.label1.Text = obj.Name;
                    f.textBox1.Text = obj.EmbeddedText;

                    if (f.ShowDialog(this) == DialogResult.OK)
                    {
                        obj.EmbeddedText = f.textBox1.Text;
                        Memo.ReformatAll(false, true);
                    }
```

```
            }
        }
        finally
        {
            f.Dispose();
        }
    }
```

4.1.1.2 S) OnShowHint

Member of WPDLLInt
Declaration C#
OnShowHint(Object Sender, int X, int Y, string Hint, **ref** bool Ignore);

Declaration OCX
OnShowHint(ByVal X As Long, ByVal Y As Long, ByVal Hint As String, Ignore As Boolean)

This event can be used to display a hint message, for example in the status bar of the application. The event will be fired when the mouse hovers one of the buttons.

```
Private void wpdllInt1_OnShowHint(Object Sender, int X, int Y, String Hint, ref bool Igno
{
   stHint.Text = Hint;
   Ignore = True;
}
```

4.1.1.2 T) OnTextObjectMouse

Member of WPDLLInt
Declaration C#
OnTextObjectMouse(Object Sender, int Editor, int State, IWPTextObj TextObj, **ref** bool Ignore);

Declaration OCX
OnTextObjectMouse(ByVal Editor As Long, ByVal State As Long, ByVal TextObj As WPTDynInt.IWPTextObj, Ignore As Boolean)

This event is used for different actions the user can perform with an object:
State = 3: the use presses the middle mouse button.
State = 2: the use presses the right mouse button.
State = 1: the use presses the left mouse button.
State = 0: the user moves the mouse over the object
State = -1: the use releases the left mouse button.
State = -2: the use releases the right mouse button.
State = -3: the use releases the middle mouse button.

Using the reference to the object TextObj you can manipulate it. To switch a text frame (TextDynamic "Premium") you can call procedure Contents_edit.

4.1.1.2 U) OnUndoChange

Member of WPDLLInt
Declaration C#
OnUndoChange(Object Sender, int Editor, int Flags)

Declaration OCX
OnUndoChange(ByVal Editor As Long, ByVal Flags As Long)

This event is triggered to let you change the enabled state of menu items which execute the Undo / Redo methods.

Parameter "Flags" is a bitfield:
bit 1 : text is selected
bit 2 : undo is possible
bit 3 : redo is possible

Category
Display Status Information

4.1.1.2 V) OnUpdateGUI

This is the main event to update any custom toolbar and menu.

Member of WPDLLInt
Declaration C#
OnUpdateGUIEvent(Object Sender, int Editor, int UpdateFlags, int StateFlags, int PageNr, int PageCount, int LineNr)

Declaration OCX
OnUpdateGUI(ByVal Editor As Long, ByVal UpdateFlags As Long, ByVal StateFlags As Long, ByVal PageNr As Long, ByVal PageCount As Long, ByVal LineNr As Long)

The event can be used to show the current position in a statusbar:

```
Private void wpdllInt1_OnUpdateGUI(Object Sender,
   int Editor,
   int UpdateFlags,
   int StateFlags,
   int PageNr,
   int PageCount,
   int LineNr)
{
      stPage.Text = Convert.ToString(PageNr)+'/'+ Convert.ToString(PageCount);
      stLine.Text = "Line " + Convert.ToString(LineNr);
      stIns.Text = (((StateFlags&2)!=0)?"INS":"");
}
```

This is also the recommended event to update any custom toolbar and menu. Inside this event You can use the **wpaGetFlags** to retrieve the states of all implemented wpa actions in one array.
C# example:

```
private void wpdllInt1_OnUpdateGUI(object Sender, int Editor, int UpdateFlags, int StateF
{
      int wpa_italic = wpdllInt1.wpaGetID("Italic");
      byte[] stateflags = wpdllInt1.wpaGetFlags(0); // Current editor
      btItalic.Pushed =(stateflags[wpa_italic] & 2)!=0;
}
```

In this example we use wpaGetID to get the ID for a certain action to make the process better to understand. In a real world application you would use wpaGetID() only once and then save the result id in an integer property added to the button or menu class. (see example "First C# Application" in the manual)

Note: You can use **wpaSetFlags** to modify certain internal states. This makes it possible to disable internal buttons under program control.

Also see: `IWPAttrInterface TextAttr;`

Parameters	
Editor	The number of the current editor, 1 or 2
UpdateFlags	The bitfield UpdateFlags selects the elements of the GUI which need update: bit 1 (1): Selection has been changed/ bit 2 (2): Undo/Redo state has been changed bit 3 (4): Paragraph properties have been changed bit 4 (8): Character attributes have been changed bit 5 (16): The Layout mode was changed bit 6 (32): Cursor position has been changed bit 7 (64): The Readonly property was changed bit 8 (128): The editor features/license has been changed Usually you do not have to use this parameter. It is internally used to update the flag array in a more effective way.
StateFlags	The bitfield StateFlags contains the flags: bit 1 (1): The text was modified bit 2 (2): The editor is insertion mode, otherwise overwrite mode bit 3 (4): Can Undo bit 4 (8): Can Redo bit 5 (16): Currently text or an object is selected bit 6 (32): Currently an object is selected bit 7 (64): The selected object is an image bit 8 (128): The cursor is undefined - the current position has not been set bit 9 (256): The cursor is within a table bit 10 (512): The current paragraph uses a style bit 12 (1024): The current position is not inside the text body but in header/ footer, text box or footnote. uses a style
PageNr	The current page number
PageCount	The currenbt page count
LineNr	The current line number on the current page

VB 6 Example:

Here we took the automatically created MDI example and converted it to use TextDynamic. The way a the toolbar is processed here is not optimal - it uses fixed names. We would recommend to use IDs in the toolbar and use a separate array to match the button to the action.

This is the code which simply uses the original logic:

```
Private Sub rtfText_OnUpdateGUI(ByVal Editor As Long, ByVal UpdateFlags As Long, ByVal St
  Dim s As String ' the returned array is a string
  Dim i As Integer ' we need the index of a certain action
  Dim Bytes() As Byte ' we need a bytes array to test the flags

  ' Retrieve the setting of all atcions
  s = rtfText.wpaGetFlags(0) ' this are the flags for the current editor
  Bytes = StrConv(s, vbFromUnicode) ' convert string to bytes

  ' Character attributes have been changed
  If (UpdateFlags And 8 <> 0) Then
    ' this is the id for the command to toggle 'bold'
    i = rtfText.wpaGetID("bold")
    ' Use the id to update the setting of 'Bold'
    fMainForm.tbToolBar.Buttons("Fett").Value = IIf(Bytes(i) And 2, tbrPressed, tbrUnpres
    ' Proceed with the other elements
```

```
    ' ...
    ' this is the id for the command to toggle 'italic'
    i = rtfText.wpaGetID("italic")
    ' Use the id to update the setting of 'italic'
    fMainForm.tbToolBar.Buttons("Kursiv").Value = IIf(Bytes(i) And 2, tbrPressed, tbrUnpr


    ' this is the id for the command to toggle 'underline'
    i = rtfText.wpaGetID("underline")
    ' Use the id to update the setting of 'underlined'
    fMainForm.tbToolBar.Buttons("Unterstrichen").Value = IIf(Bytes(i) And 2, tbrPressed,
  End If

  ' Paragraph attributes have been changed
  If (UpdateFlags And 4 <> 0) Then
    ' this is the id for the command to toggle 'left'
    i = rtfText.wpaGetID("left")
    ' Use the id to update the setting of 'left aligned'
    fMainForm.tbToolBar.Buttons("Links ausrichten").Value = IIf(Bytes(i) And 2, tbrPresse
    ' Proceed with the other elements
    ' ...
    ' this is the id for the command to toggle 'Center'
    i = rtfText.wpaGetID("Center")
    ' Use the id to update the setting of 'Centered'
    fMainForm.tbToolBar.Buttons("Zentrieren").Value = IIf(Bytes(i) And 2, tbrPressed, tbr


    ' this is the id for the command to toggle 'Right'
    i = rtfText.wpaGetID("Right")
    ' Use the id to update the setting of 'Rechts ausrichten'
    fMainForm.tbToolBar.Buttons("Rechts ausrichten").Value = IIf(Bytes(i) And 2, tbrPress
  End If

End Sub
```

Category
Display Status Information

4.1.1.2 W) OnKeyDown

Member of WPDLLInt

Declaration C#
KeyDown(KeyEventArgs e)

Here the standard event is used.

You need to cast e to wpKeyEventArgs which contains the properties Editor, Shift, Alt and Control.

```
private void wpdllInt1_KeyDown(object sender, System.Windows.Forms.KeyEventArgs e)
{
    if(((wpKeyEventArgs)e).Control)
            MessageBox.Show("Control");
}
```

Declaration OCX

OnKeyDown(ByVal Editor As Long, Key As Integer, ByVal Shift As Long)

This event is triggered when the user presses a key on the keyboard.

Parameters

Editor: The editor number, 1 or 2
Key: The key board number as VK_ value
Shift: A bit field representing the state of the control keys:
1 : Shift key
2 : ALT key
4 : Ctrl key

Example VB6:

```
Private Sub WPDLLInt1_OnKeyDown(ByVal Editor As Long, Key As Integer, ByVal Shift As Long
  If Shift = 4 Then
      ' Ctrl+B
      If Key = Asc("B") Then
            WPDLLInt1.TextAttr.ToggleStyle (0)
      ' Ctrl+I
      ElseIf Key = Asc("I") Then
            WPDLLInt1.TextAttr.ToggleStyle (1)
      ' Ctrl+U
      ElseIf Key = Asc("U") Then
            WPDLLInt1.TextAttr.ToggleStyle (2)
      End If
  End If
End Sub
```

Trouble shooting:
If your code does not work please check if you have a global short cut which reverts the change.

4.1.1.2 X) OnKeyPress

Member of WPDLLInt
Declaration C#
KeyPress(KeyPressEventArgs e)

Here the standard event is used.
You need to cast e to wpKeyEventArgs which contains the properties Editor, Shift, Alt and Control:

```
if ((wpKeyEventArgs)e).Control) ...
```

Declaration OCX
OnKeyPress(ByVal Editor As Long, Key As Byte)

This event is triggered when the user types on the keyboard. It receives the key as character value.
You can use this event to implement short cuts, for example activate 'bold' when Ctrl+B is pressed:

Example VB6 (also see OnKeyDown):

```
Private Sub WPDLLInt1_OnKeyPress(ByVal Editor As Long, Key As Byte)
  Dim Memo As IWPMemo
```

```
  If Editor = 2 Then Set Memo = WPDLLInt1.Memo2 Else: Set Memo = WPDLLInt1.Memo
  If Key = 2 Then ' Ctrl B
      If Memo.TextCursor.IsSelected Then
         Memo.CurrSelAttr.ToggleStyle (0)
      Else
         Memo.CurrAttr.ToggleStyle (0) ' set bold!
      End If
      Key = 0
  End If
End Sub
```

4.1.1.2 Y) OnKeyUp

Member of WPDLLInt
Declaration C#
KeyUp(KeyEventArgs e)
Here the standard event is used. You can cast e to wpKeyEventArgs which contains the properties Editor, Shift, Alt and Control.

Declaration OCX
OnKeyUp(ByVal Editor As Long, Key As Integer, ByVal Shift As Long)

This event is triggered when the user releases a key on the keyboard.

Parameters

Editor: The editor number, 1 or 2
Key: The key board number as VK_ value
Shift: A bit field representing the state of the control keys:
1 : Shift key
2 : ALT key
4 : Ctrl key

4.1.1.2 Z) OnLeaveEditor

Member of WPDLLInt
Declaration C#
Leave(EventArgs e)
Here the standard event is used. You can cast e to wpEventArgs which contains the property Editor.

Declaration OCX
OnLeaveEditor(ByVal Editor As Long)

4.1.1.2 AA) OnCompleteWord

Member of WPDLLInt
Declaration C#
OnCompleteWordEvent(Object Sender, int Editor, **ref** byte LastChar);

Declaration OCX
OnCompleteWord(ByVal Editor As Long, LastChar As Byte)

This event allows it to implement auto completion and makro functionality. If the user types in a word and then presses a word delimiting character, this event is triggered. The delimiting character will be passes as parameter LastChar. You can set it to 0 if you do not want to use it in the text, for example if the character is a control key.

To read or update the last word use SelText. You can also use LoadFromString to insert a block

of formatted text at cursor position.

VB Example:

```
Private Sub WPDLLInt1_OnCompleteWord(ByVal Editor As Long, LastChar As Byte)
  Dim w As String
  w = WPDLLInt1.CurrMemo.SelText
  If w = "mfg" Then
    WPDLLInt1.CurrMemo.SelText = "Mit freundlichen Grüßen"
  ElseIf w = "logo" Then
    WPDLLInt1.TextCursor.InputImage "c:\logo.png", 0
  End If
End Sub
```

4.1.1.2 AB) OnEnterEditor

Member of WPDLLInt
Declaration C#
Enter(EventArgs e)
Here the standard event is used. You can cast e to wpEventArgs which contains the property Editor.

Declaration OCX
OnEnterEditor(ByVal Editor As Long)

This event procedure is called after the cursor entered one of the internal editors.

4.1.2 Categories

Character Attributes	Methods to read and write the character attributes
Character Styles	Methods to read and write the attributes, bold, italic, underlined ...
Paragraphstyle Support	Interfaces, Properties and Methods to use Paragraph styles
Load and Save	Methods to load and save the text as file, stream or string. (RTF, XML, HTML ...)
Hyperlinks and Bookmarks	Create and use Hyperlinks and Bookmarks
Image Support	Methods to insert and manipulate images
Hader and Footer Support	Create and manage header and footer texts
Callback Functions	Enumerate all paragraphs, styles or header/footer texts
Table Support	Create and modify tables
Mailmerge	Update and read out data fields in the text
TextDynamic CSS strings	Use 'WPCSS' strings to store and apply attributes
Document Properties	Save and Load string variables with documents
Attribute IDs	Methods with work with attributes referenced by IDs
Position Markers	Methods to temporarily save cursor position
Lowlevel Paragraph IDs	Paragraph IDs can be used to change "current" paragraph
Logical MDI Support	Manage multiple documents in one control.
Printing	Properties and methods for printing
Reporting	With the TextDynamic report a text file (RTF or WPT format) is created which allows full editing, including changing of the page size.

4.1.2.1 Character Attributes Category

Methods to read and write the character attributes

Description
The TextDynamic word processing engine stores the attributes of characters in attribute records. This records hold 16 different attributes. To reduce memory consumption the text only contains the reference, the index value, of the attribute record. You can use the GetCharAttr function to read the index of the character at a certain position in a paragraph. The interface IWPAttrInterface is used to translate between index valuse and attributes, such as the font size or -name.

Properties

WPDLLInt.AttrHelper
IWPAttrInterface.CharAttrIndex

Methods

IWPFieldContents.FieldAttr
IWPParInterface.CharAttr
IWPParInterface.GetCharAttr
IWPParInterface.ReplaceCharAttr
IWPParInterface.SetCharAttr
IWPTextCursor.WordCharAttr

4.1.2.2 Character Styles Category

Methods to read and write the attributes, bold, italic, underlined ...

Description
The character styles are saved as bits inside the character attribute record. TextDynamic supports the flags bold, italic, underline, strikeout, super script, sub script, hidden, uppercase, lowercase, no-proof, double strikeout and protected. Two flags are reserved.

Inside the attribute record two properties are taken for all this flags, one bitfield to enable/ disable the attribute (WPAT_CharStyleON=7), a second attribute to check that a certain flag is used (WPAT_CharStyleMask=6). The methods in this category hide the complicated handling from the developer.

Since the properties all have an "undefined" state they are read and written using methods. The read method (Get) returns true if the property was set.

Read Propety:

GetFontface
GetFontSize

Write Property

SetFontface
SetFontSize

Other Methods

IWPAttrInterface.ExcludeStyles
IWPAttrInterface.GetStyles
IWPAttrInterface.IncludeStyles
IWPAttrInterface.SetStyles
IWPAttrInterface.ToggleStyle

4.1.2.3 Document Properties Category

Save and Load string variables with documents

Description
It is possible to store an unlimited count of string variables with the document. The varaiables are called "RTFVariables" although they can also be saved in WPT format.

When working with HTML files the variable "title" will be used to fill the <title> tag.

Properties	Methods
none	IWPMemo.GetRTFVariable IWPMemo.SetRTFVariable

4.1.2.4 Callback Functions Category

Enumerate all paragraphs, styles or header/footer texts

Description
TextDynamic makes it easy to retrieve information about the text and its elements. There are several events which are triggered by certain enumerate functions, so you can extract all styles in the document, format all paragraphs and check which header or footer texts are defined.
The procedure AddTable triggers an event OnCreateNewCell which is triggered for each new cell created by this methods. You can easily add text or attributes inside this event.
All callback functions get an additional user integer variable "EventParam" which has been initialized when calling the enumerate function. You can use this variable to select different algorithms in one event procedure.

Events	Methods
OnCreateNewCell	IWPFieldContents.AddTable
OnEnumDataBlocks	IWPMemo.EnumDataBlocks
OnEnumParOrStyle	IWPMemo.EnumParagraphs
OnEnumTextObj	IWPMemo.EnumParSiblings
	IWPMemo.EnumParStyles
	IWPMemo.EnumSelParagraphs
	IWPMemo.EnumTextObj
	IWPTextCursor.AddTable

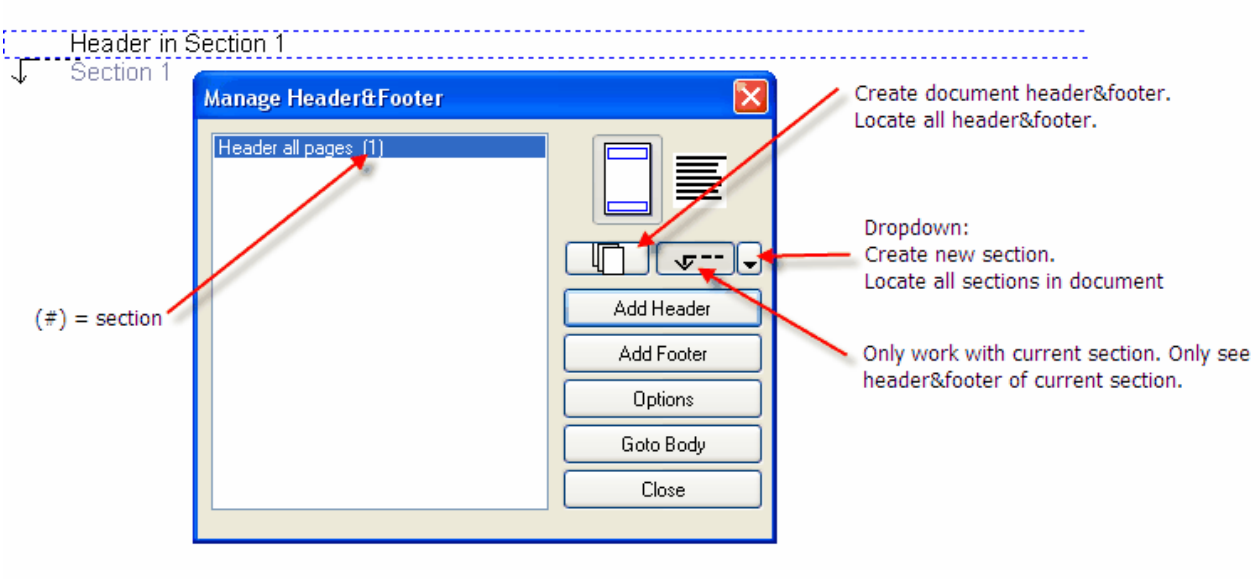
4.1.2.5 Header and Footer Support Category

Create and manage header and footer texts

Description
The support for header and footer texts is exceptionally strong in TextDynamic. It is possible to have a different header and footer on each single page in the document. Each of this header and footer texts can be edited WYSIWYG, the user only has to click into the header/footer area. It is also possible to create a header or footer when the user double clicks into the margin.

The event OnGetSpecialText can be used to select custom header and footer. Otherwise "ranges" are used to select the text. Possible ranges include "only on first page", "on odd pages", "on even pages" or "on all pages".

The dialog "**Manage header&footer**" is used to find and select all header and footer.



See Properties of
IWPDataBlock

Events
OnClickCreateHeaderFooter
(not in RTF2PDF)
OnGetSpecialText

Methods
IWPMemo.BlockAdd
IWPMemo.BlockAppend
IWPMemo.BlockFind
IWPMemo.FindFooter
IWPMemo.FindHeader
IWPTextCursor.GotoBody
IWPTextCursor.InputFooter
IWPTextCursor.InputHeader

4.1.2.6 Sections

Create a Section / Examine the current section
IWPTextCursor.InputSection

Access Section
IWPPageSize

Read Section ID:
IWPPageSize.GetProp(0)

This methods are useful to manage header/footer
BlockAdd
BlockAppend
BlockFind

4.1.2.7 Hyperlinks and Bookmarks Category

Create and use Hyperlinks and Bookmarks

Description
Hyperlinks are text parts which are embedded into objects - just like in HTML: <a>this is a link. Bookmarks use the same technique, only the objects (see IWPTextObj) have a different object type ObjType.

When the user click on the text which has been marked to be a hyperlink the event OnHyperlink will be triggered. You can now load a different document or locate a bookmark.

Event	Methods
OnHyperlink	Memo.SpecialTextAttr IWPFieldContents.InputHyperlink IWPMemo.GetObjAtXY IWPTextCursor.InputBookmark IWPTextCursor.InputHyperlink IWPTextCursor.MoveToBookmark

Tip: You can use IWPMemo.EnumTextObj to read the url and text of all hyperlinks.

Also see: Example: Convert to HTML

4.1.2.8 Image Support Category

Methods to insert and manipulate images

Description
Like Hyperlinks and merge fields the images are inserted into the text as "text objects". But in contrast to the objects named first, images use a data object which is attached to the "text object". This makes it possible to use the same image data for different text objects which each show the same image, possibly at a different size - (IWPTextObj.SetContentsID).

In all cases the IWPTextObj interface is used to read and write the properties of the reference object.

Events	Methods
OnAfterSaveImage OnBeforeSaveImage	Memo.InsertGraphicDialog IWPParInterface.InsertNewObject IWPTextCursor.InputImage IWPTextCursor.InputPicture IWPTextObj.LoadFromFile IWPTextObj.SetContentsID
Interface to the object placeholder IWPTextObj	

4.1.2.9 Load and Save Category - Formatstrings

Methods to load and save the text as file, stream or string.

Description
TextDynamic contains various methods to load and save text. Most of these methods require a second parameter, the "format string". This string first contains the required format RTF, ANSI, HTML, WPT, UNICODE, MIME and then options for the reader or writer.

New: Before the format name it is now possible to specify the **name of a merge field** or other paired objects. The syntax is X:*text*=.
X can be one of this letters:
lowercase letters:

- c: search for wpobjCustom comparing the object name = *text*
- f: search for wpobjMergeField
- h: search for wpobjHyperlink
- b: search for wpobjBookmark

p: search for wpobjTextProtection
s: search for wpobjSPANStyle
capital letters:
C: search for wpobjCustom comparing the object source parameter = *text*
F: search for wpobjMergeField etc.
If the objects are not found nothing will be loaded or saved.

Properties

IWPDataBlock.Text
IWPMemo.LastFileName
IWPMemo.Modified
IWPMemo.SelText
IWPMemo.Text

Methods

IWPFieldContents.LoadText
IWPMemo.GetPageAsMetafile
IWPMemo.InsertGraphicDialog
IWPMemo.Load
IWPMemo.LoadEx
IWPMemo.LoadFromFile
IWPMemo.LoadFromStream
IWPMemo.LoadFromString
IWPMemo.LoadFromVar
IWPMemo.Save
IWPMemo.SaveAs
IWPMemo.SavePageAsMetafile
IWPMemo.SaveToFile
IWPMemo.SaveToStream
IWPMemo.SaveToString
IWPMemo.SaveToVar
IWPParInterface.LoadFromFile
IWPParInterface.LoadFromString
IWPParInterface.SaveToString
IWPPdfCreator.Print
IWPTextObj.LoadFromFile
IWPTextObj.LoadFromStream

Many methods which save text also use a boolean value to toggle between saving the complete text/selection only.

The loading methods also use a parameter to select the insert at "cursor position" mode.

Please also note that it is possible to load inside a paragraph or cell, and it is also possible to load the text into the "data block" which represents a header, footer, text box or footnote.

If you load and save to a blob field we recommend to use GetTextVar / SetTextVar. This works much faster than using strings and avoids problems which are caused by unicode compression. GetTextVar can be also used to read the binary data which should be used for an ASP Response.

When using streams in .NET applications you need the **`Stream2WPStream`** utility class. It needs to be created for each load and save operation.

```
System.IO.Stream str = new System.IO.MemoryStream();
Memo.SaveToStream( new WPDynamic.Stream2WPStream(str) ,false,"RTF");
str.Position = 0;
Memo.LoadFromStream(new WPDynamic.Stream2WPStream(str), true, "AUTO");
```

List of options possible in a "format string":

Option "-..."	What it does	RTF READ	RTF WRITE	HTML READ	HTML WRITE	WPT READ	WPT WRITE	ANSI *.txt	UNICODE	MIME *.msg

codepage 12xx	set the code page XY for loading. Default is 1252	X		X				X		
utf8	The HTML reader will detect UTF8 characters			X						
onlybody	only load/write body part (no header + footer)		X		X	X	X			
ignorefonts	do not load font information	X				X				
ignorefontsize	do not load font size information	X				X				
nostyles	do not load or save styles	X	X			X	X			
nonumstyles	do not load or save number styles	X	X			X	X			
noimages	do not load or save images	X	X	X		X	X			
basetext	only save text in WPTOOLS format, no styles or other info									
nomergefields	do not save merge fields, only contents									
nohyperlinks	do not save hyperlinks									
nobookmarks	don't save bookmarks									
novariables	dont save or load "RTFVariables" (also known as "Document Properties")	X	X				X			
nobinary	try to convert all binary to ASCII		X							
nopageinfo	do not save or load page size information		X			X	X			
ignorekeepn	don't load the keepn flag from RTF	X	X			X	X			
ignorerowmerge	do not load row merging from RTF	X								
ignorecollapsedpar	don't save table rows which are hidden by WPReporter		X							
dontaddrtfvariables	only load those content of RTF variables which are present in collection	X				X				
complete	load header and footer information although we are inserting text									
alwaysembed	embed image data also for linked images									
nonumberprops	the numbering will be saved as normal text. Number styles will not be saved either.		X							
nospanstyles	span styles will not be saved as objects -only the properties will be saved.		X							
nocharstyles	don't write character styles (\cs)		X							
ignorerowme	If this flag is set the	X								

rge	rowmerge property will be ignored when reading RTF code.									
dontfixattr	If this property is true the RTF reader will **not** remove redundant attribute information, such as fonts, indents etc. Redundant are attributes which are selected by the paragraph styles or document font information (DefaultTextAttr).	**X**								
ignorerowheight	Do not load and apply the absolute row height defined in RTF code.	**X**								
onlyusedstyles	If this property is TRUE the RTF reader will only add the styles which are used.	**X**								
onlystandardheadfoot	If this property is true the RTF writer will not save header and footer texts which rage is not defined by the RTF specifications, such as 'on last page'.		**X**							
NoStartTags	The RTF and HTML writer do not create the standard RTF and HTML opening and closing sequence		**X**							
abbreviated	The WPTOOLS writer created abbreviated property codes						**X**			
nosectiondata	The WPTOOLS writer does not write section information						**X**			
ignorehtml	If this property is true the HTML reader will not use HTML tags. This is useful if also useBBCodes is used.			**X**						
usebbcodes	If this property is tre the HTML will convert BB codes into HTML properties			**X**						
nospanobjects	This property is true SPAN tags will not be embedded.			**X**						
usecr	If this property is true CR codes in the HTML text will be used as paragraph breaks			**X**						
imgpath="..."	Specify the path where to create linked files				**X**					

	(default = "\" which is the current directory)									
csspath="..."	Specify the location of a CSS file which is used instead of the one linked by the HTML code when reading. When writing the respective link will be written.			X	X					
dontwritestyleparam	Do not write the style="" parameter in HTML code. This creates simplified HTML code.				X					
writeallcolwidth	Write the column width for all table columns				X					
writebasefont	Write a basefont tag using the information of the default attributes of the document				X					
writespan	Write objects instead of <font>				X					
littleendian	Write little endian unicode file								X	
nohtml	do not save HTML text									X
noplain	do not save PLAIN text									X
readprintpar	the paper source in in IWPPrintParameter will be loaded from RTF	X								

4.1.2.10 Mailmerge Category

Update and read out data fields in the text. (See Introduction)

Description
The mailmerge process offered by TextDynamic is one of its most popular features.

It updates the text inside certain field objects. It can also update image objects.

Since the fields are usually not deleted by the update process it is possible to update the data later again. Mail merge is more than just search & replace.

It is also possible to read out the text in case the user has edited it. Like hyperlinks merge fields use paired objects. One object marks the start of the field, the other the end. The text within is the "embedded text".

You can customize the component to show or hide the field markers (`SpecialTextAttr(wpInsertpoints).Hidden`). It is also possible to display a single object only which shows the name of the field. (`property IWPMemo.ShowFields`)

The merge process is started with IWPMemo.MergeText("").
You need to create an event handler for OnFieldGetText to insert the data into the field.

Mail merge fields are implemented using paired text objects. Using InputField and

CurrObj.Mode=2 you can create an "Edit Field" which can be changed in form completion mode while the rest of the text is protected.

Properties	Methods
IWPFieldContents.StringValue IWPMemo.LabelDef IWPMemo.ShowFields	IWPFieldContents.ContinueOptions IWPMemo.AppendOtherText IWPMemo.DeleteParWithCondition IWPMemo.MergeText IWPMemo.RTFDataAppendTo IWPTextCursor.FieldsFromTokens (akna. "ReplaceTokens") IWPTextCursor.InputField IWPTextCursor.MoveToField

IWPMemo.CurrObj this is the field which was added using InputField last.

The **event** OnFieldGetText receives a reference to IWPFieldContents which contains many properties.

Note: The field markers « and » are usually visible. To hide the markers use

```
Memo.SpecialTextAttr(wpInsertpoints).Hidden = True
```

Also see:
Command ID 16 - Syntax Highlighting - highlight tokens, i.e. **<<name>>**

Command ID 17 - Token To Template Conversion - convert tokens to fields

Command ID 15 - Update ShowFields Mode - display field names instead of « and ».

4.1.2.11 Position Markers Category

Methods to temporarily save cursor position

Description
The markers can be used to save the position of the cursor temporarily so you can return to that position without any delay. Markers will not change the text and do not survive load&save operations.

Most insert methods adjust the markers automatically, so they work better than simply storing and assigning the "CPPosition".

Properties	Methods
none	IWPTextCursor.MarkerCollect IWPTextCursor.MarkerCollectAll IWPTextCursor.MarkerCPPosition IWPTextCursor.MarkerDrop IWPTextCursor.MarkerGoto IWPTextCursor.MarkerSelect

If you just need to move back or forward in the text please also do not use "CPPosition" but one of these methods from IWPTextCursor:

CPMoveAfterTable	**CPMoveNextRow**
CPMoveBack	**CPMoveNextTable**
CPMoveBeforeTable	**CPMoveParentTable**
CPMoveNext	**CPMovePrevCell**

CPMoveNextBand
CPMoveNextCell
CPMoveNextGroup
CPMoveNextObject
CPMoveNextPar
CPMovePrevObject
CPMovePrevPar
CPMovePrevRow
CPMovePrevTable

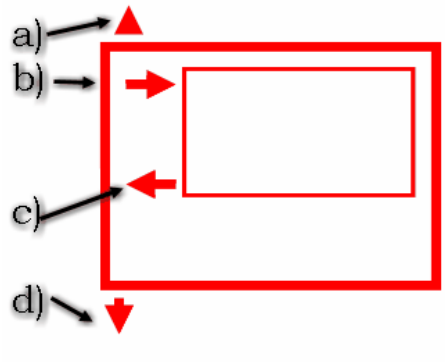
4.1.2.12 Lowlevel Paragraph IDs Category

Paragraph IDs can be used to change "current" paragraph

Description

When working with the IWPParInterface, Paragraph IDs make it possible to switch between Next, Previous, Children and Parent Paragraphs.

So you can actually use the property CurrPar to access, evaluate and modify all paragraph in the internal XML like paragraph tree.



Get ID:
IWPParInterface.GetPtr
Set ID:
IWPParInterface.SetPtr

a) previous in list (i.e. previous paragraph, cell to the left)
IWPParInterface.GetPtrPrev

b) first child or 0 (i.e. first cell in table row, first row in table, second paragraph in cell)
IWPParInterface.GetPtrChild

c) parent or 0 (i.e parent row of cell, parent table of row)
IWPParInterface.GetPtrParent

d) next in list (i.e. next paragraph, cell to the right)
IWPParInterface.GetPtrNext

Please do not save the Ptr value to be used at a later time in the application. This can cause severe problems when the paragraph was deleted.

Please also see the low level move methods - they provide a save way to loop all paragraphs in the current document or text block.

4.1.2.13 Printing Category

Properties and methods for printing

Properties
IWPMemo.PageSizeList
IWPMemo.PrintParameter

Methods
Select Printer
Memo.TextCommandStr(10, name).
or

Start Printing:
Memo.TextCommandStr(11, range).
or **IWPMemo.Print** and

IWPMemo.PrintPages

Create printing cue:

Alternatively you can use
Memo.TextCommandStr 13
and Memo.TextCommandStr ID 14

to use the optional PDF export:
IWPPdfCreator.Print

Also see:
Memo.TextCommandStr ID 34 -Print To File

Tip:
When using **RTF2PDF / TextDynamic Server** in DLL mode (not using the COM interfaces) this command IDs are available:

```
WPCOM_SELECT_PRINTER = 1320;  // param = printer name
WPCOM_PRINT  = 1321;          // param = page list
WPCOM_PRINT2 = 1322;          // print memo 2, param = page list
WPCOM_BEGIN_PRINT = 1323;     // when printing multiple documents into one printing cue
WPCOM_END_PRINT = 1324;       // use beginprint/endprint
WPCOM_SET_PRINTFILE = 1325; // set print file for subsequent printing
```

4.1.2.14 Standard Editing Commands Category

Commands usually found in menu "Edit"

Description
This category contains links to commands to update the selection of text, for undo and redo.

Properties

Methods

IWPMemo.CopyToClipboard
IWPMemo.CutToClipboard
IWPMemo.PasteFromClipboard
IWPTextCursor.EnabledProtection
IWPTextCursor.SelectAll
IWPTextCursor.SelectLine
IWPTextCursor.SelectParagraph
IWPTextCursor.SelectTable

4.1.2.15 Paragraphstyle Support Category

Interfaces, Properties and Methods to use Paragraphstyles

Description
Paragraphstyles are very useful to give your documents a uniform look. Instead of applying the attributes font size=12, underlined to each paragraph which contains a headline, you can simply attach the style "headline" to that paragraph. The style headline is defined separately and may be modified any time. After the modification (and a "ReformatAll" of the text) the new style will be displayed.

In general, when you attach a style the original attributes of neither the paragraph nor the characters will be changed, there is simply a new set of default attributes which is valid only for this paragraph. When the engine is formatting the text the default attributes are overruled by the attributes defined in the paragraph itself and the character attributes. So, to make sure that all attributes defined in a style are actually used, the attributes must not be used by the paragraph or characters.
(Note: The same logic is used by HTML/CSS - Attributes can be defined on a style, paragraph and character basis)

The method which assigns a style to a paragraph will automatically remove all attributes from the paragraph and the characters which would hide (overrule) the attributes defined in that style.

Properties

IWPMemo.CurrStyle
IWPMemo.CurrStyleAttr
IWPParInterface.StyleName
IWPTextCursor.CPStylePtr
IWPTextObj.StyleName

also see:
IWPParInterface.ParStrCommand

Methods

IWPMemo.DeleteStyle
IWPMemo.EnumParStyles
IWPMemo.SelectStyle
IWPTextCursor.InputParagraph
Memo.LoadNumberStyles
Memo.SaveNumberStyles
Memo.LoadStyleSheet
Memo.SaveStyleSheet

Command ID 28/29/30/31 - Save the paragraph styles into a file or string

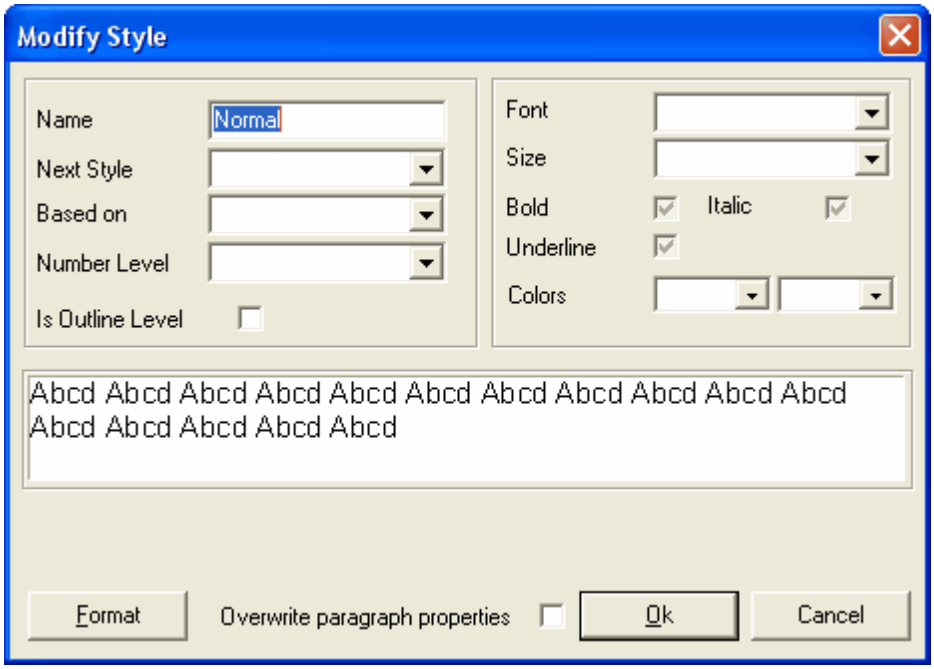
Also see SetXMLSchema and "Paragraph Styles".

GUI:
In the XML layout description in the PCC file this line will display a combo box with paragraph styles:

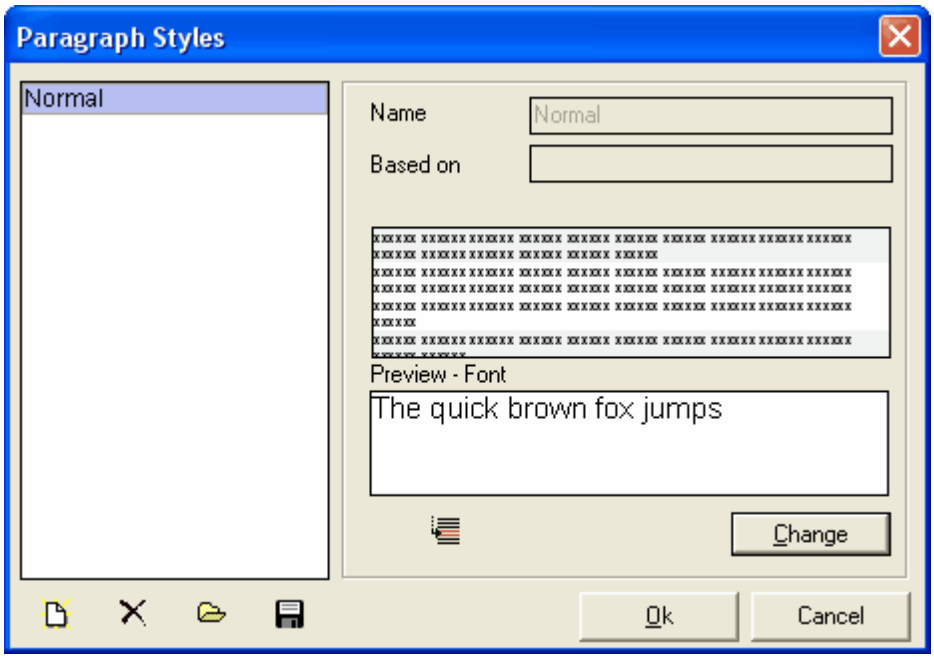
```
<dropdownlist width="150" Image="-1" text="" wpa="ParStyleSelection"/>
```

WPA Actions:

DiaOneStyle - edit one style



DiaStyleSheet - edit all styles



ParStyleSelection

- select the paragraph style list in the toolbar (see example above)

4.1.2.16 Table Support Category

In TextDynamic tables are handled the same as HTML table. On table object contains table rows which contain table cell objects. In TextDynamic all this objects internally are paragraph objects, only the property ParagraphType selects a different mode. This makes it possible to replace an empty paragraph with a table by simply changing its mode and adding row and cell children to it.

TextDynamic supports 4 different ways to create a table by code. We offer so many possibilities because the data which has to be placed into the table cells can come in different

ways and order. In general we favourize the use of a callback to fill the cell text, but it is not always possible to callbacks. The possibility to select from different methods makes it easy to adapt existing logic to work with TextDynamic.

Method 1 - use callback:
Call TextCursor.AddTable and use the event OnCreateNewCell to format and fill the cell. If you do not know the count of rows in advance pass 100000 and abort the creation loop inside the OnCreateNewCell event using the variable parameter "AbortAtRowEnd".

Tip: Alternatively to the callback you can use the method ASetCellProp to modify a group of cells after the complete table was created.

Advantage: Table is created automatically, only cells which need modification need 'attention'.
Disadvantage: Callback function can be difficult to read and maintain. Sometimes callback is not possible (script languages)

Method 2 - simulate user input:
Call TextCursor.AddTable - then use the properties CPTableRowNr, CPTableColNr to "move around" and insert text using InputString. You can also move the cursor using CPMoveNextRow, CPMoveNextCell which would be faster.

Advantage: Table is created automatically, only cells which need modification need 'attention'.
Disadvantage: can be slow with large tables

Method 3 - create from top to bottom:
Use TextCursor.InputTable, then InputRowStart, as many InputCell ass needed and InputRowEnd to close the current row. Create new row with InputRowStart and so on.

Advantage: fast, easy to understand logic
Disadvantage: can create rows with uneven count of columns

Method 4 - work with objects:
Create a new paragraph or modify Memo.CurrPar to make it a table object: par.SetParType((int) ParagraphType.Table). Now you can use AppendChild to create a new row and for each row use AppendChild to create a new cell. To process a different paragraph you can either use the Select methods, or you save the paragraph ID and use par.SetPtr(id).

Please see the example code in topic IWPDataBlock.AppendParagraph.

Advantage: Table can be created without change of cursor position
Disadvantage: Difficult to understand, can create rows with uneven count of columns. Exceptions are possible when SetPtr() is not correctly used.

Properties	Methods	Methods
IWPParInterface.CellCommand	IWPDataBlock.AppendParagraph	IWPTextCursor.CPMoveNextCell
IWPParInterface.CellName	IWPFieldContents.AddTable	IWPTextCursor.CPMoveNextRow
IWPParInterface.IsColMerge	IWPMemo.Tables_WidthFixed	IWPTextCursor.CPMoveNextTable
IWPParInterface.IsFooterRow	IWPMemo.Tables_WidthPC	IWPTextCursor.CPMoveParentTable
IWPParInterface.IsHeaderRow	IWPParInterface.AppendChild	IWPTextCursor.InputCell
IWPParInterface.	IWPParInterface.SetParType	IWPTextCursor.InputRowEnd
	IWPReportBand.AddTable	IWPTextCursor.InputRowStart
	IWPTextCursor.AddTable	IWPTextCursor.InputTable
	IWPTextCursor.AppendRow	IWPTextCursor.SelectTable

IsRowMerge
IWPParInterface.WidthTW
IWPTextCursor.CPTableColNr
IWPTextCursor.CPTablePtr
IWPTextCursor.CPTableRowNr

IWPTextCursor.CombineCellHorz
IWPTextCursor.CombineCellVert
IWPTextCursor.CPMoveAfterTable
IWPTextCursor.CPMoveBeforeTable

IWPTextCursor.SelectTableColumn
IWPTextCursor.SelectTableRow
IWPTextCursor.SetTableLeftRight
IWPTextCursor.TableClear
IWPTextCursor.TableDelete

Event
OnCreateNewCell

Methods to quickly modify an existing table without user interaction:
IWPTextCursor.ExitTable
IWPTextCursor.TableSplit
IWPTextCursor.TableSort
IWPTextCursor.ASetCellProp
IWPTextCursor.MergeCellHorz
IWPTextCursor.MergeCellVert
IWPTextCursor.ASetCellStyle

Also available:

TextCommand(13) combines all adjacent tables. Returns the count of deleted tables.

TextCommand(14) tries to create merged cells for cells which are wider than the others in a column. Returns the count of changed tables.

4.1.2.17 TextDynamic CSS strings Category

Use 'WPCSS' strings to store and apply attributes

Description
Paragraph and character attributes used by paragraphs, text styles and characters can be loaded and saved in a special format which can be save to and loaded from a string. So you only need one line of code to copy the attributes of one paragraph into a text style, or save it for later use.

TextDynamic also supports standard CSS style descriptions but this format is not capable to hold all possible attributes, i.e. tabstops.

Properties	Methods
IWPParInterface.ParCSS	IWPAttrInterface.GetWPCSS
IWPParInterface.ParWPCSS	IWPAttrInterface.SetWPCSS

Also see:
SelectStyle
LoadNumberStyles
SaveNumberStyles
LoadStyleSheet
SaveStyleSheet

4.1.2.18 Tabstop Category

In the **Interface IWPParInterface** You can use this methods

IWPParInterface.TabstopAdd
IWPParInterface.TabstopClear
IWPParInterface.TabstopDelete
IWPParInterface.TabCount
IWPParInterface.TabGet
IWPParInterface.TabMove
IWPParInterface.TabFind
IWPParInterface.TabGetNext

To work with the tabs in the selected text please use
Memo.TextCommand(ID,...)

ID: 21
Adds a tab stop. This method works with selections or the current paragraph.
paramA = position in twips
paramB, low byte, the mode (left, right, center, decimal)
paramB, high byte, the fill mode (no fill, dots, MDots, hyphen, underline, THyphen, EqualSign, Arrow)

ID: 22
Deletes the tab stop at position paramA

ID: 23
Clears all tabs tops in the selected text or the current paragraph.

4.1.3 IWPMemo / IWPEditor

Interface to work with editor #1 or editor #2

Description
This interface gives access to important properties of each editor in the TextDynamic control.

The interface *IWPEditor* publishes sub interfaces to change the font and paragraph attributes of the text (TextAttr, CurrAttr, CurSelAttr), to move the cursor and insert text or data (TextCursor). It includes methods to load and save the text (LoadFromFile ...) and properties to change the layout and display. (LayoutMode, WordWrap, AutoZoom).

It contains methods to enumerate the paragraphs and styles, to work with header and footers or images.

It also contains the method SetBProp which is used to change various options of the editor for viewing and formatting.

You can also activate XML, or merge field syntax highlighting!

IWPEditor also publishes other interfaces, such as TextCursor. This interface contains the method to move the cursor and to insert and create text, text boxes and tables.

You can use Memo.Statistic to retrieve information about the count of words, lines and pages.

Note: The editor component TextDynamic uses the interface IWPMemo, the product wRtf2PDF / TextDynamic Server which was optimized for server used (such as, but not limited to ASP and ASP.NET)uses IWPEditor. In both cases the property Memo and Memo2 are used to publish the interfaces. The interface IWPEditor contains a subset of the possibilities of IWPMemo.

Since the RTF2PDF Version 3.55 the .NET assemblies publishes this interface as native .NET class. This improves the performance improves the integration into the .NET frame work, i.e. some of the methods have been overloaded to accept .NET streams.

Properties	Methods	Methods
ActiveText		
AutoZoom	AppendOtherText	
ColorDesktop	BlockAdd	
ColorHighlight	BlockAppend	
ColorHighlightText	BlockFind	
ColorPaper	Clear	
CurrAttr	ClientToScreen	
CurrBand	CopyToClipboard	
CurrentZooming	CutToClipboard	
CurrGroup	DebugShowParProps	
CurrObj	DeleteLeadingSpace	
CurrPar	DeletePage	
CurrParAttr	DeleteParWithCondition	
CurrSelAttr	DeleteStyle	
CurrSelObj	DeleteTrailingSpaces	
CurrStyle	EnumDataBlocks	
CurrStyleAttr	EnumParagraphs	
DefaultIOFormat	EnumParSiblings	
EditingMode	EnumParStyles	
Focussed	EnumSelParagraphs	
FormCompletion	EnumTextObj	
GUIMode	FindFooter	
Hidden	FindHeader	
InitialDir	GetMouseXY	
LabelDef	GetNumberStyle	
LastFileName	GetObjAtXY	
LayoutMode	GetPageAsMetafile	
Modified	GetPosAtXY	
PageSize	GetRTFVariable	
PageSizeList	GetXY	
PrintParameter	InsertGraphicDialog	
Readonly	Load	
SelText	LoadEx	
ShowFields	LoadFromFile	
Text	LoadFromStream	
TextCursor	LoadFromString	
TextReadFormat	LoadFromVar	

TextWriteFormat
TopOffset
WordWrap
Zooming
DefaultAttr

4.1.3.1 Properties

4.1.3.1 A) ActiveText

Declaration
```
IWPDataBlock ActiveText;
```

Description
This is a IWPDataBlock reference to the current (active) text layer. This can be the body text or a header or footer text. If you have a premium license this can be also the text of a footnote or the text inside of a text box. To make a certain text block active set its property WorkOnText to true.

4.1.3.1 B) TextAttr

Declaration
```
IWPAttrInterface TextAttr;
```

Description
This is the interface to the change the current writing (CurrAttr) attribute of the active editor unless text is selected.

In case text has been is selected the properties of the selected text (CurrSelAttr) will be updated.

To change paragraph attributes please use CurrPar.

This property is used best to apply and read the text attributes to implement a custom toolbar.

Example:
```
private void wpdllInt1_OnUpdateGUI(
      object Sender,
      int Editor,
      int UpdateFlags,
      int StateFlags,
      int PageNr,
      int PageCount,
      int LineNr)
{
         string n = "";
         wpdllInt1.Memo.TextAttr.GetFontface(ref n);
         FontCombo.Text = n;
}
```

If a function of TextAttr returns FALSE this means the default attribute is used for this property.
The default text attributes can be changed in Memo.DefaultAttr.

IMPORTANT:

Memo.TextAttr provides either a reference to CurrAttr or to CurrSelAttr, depending on the state of TextCursor.IsSelected. Please note, that an initialization before the selection process will cause the wrong interface to be provided.

This will code not modify the attribute of "some text":

```
IWPMemo Memo = wpdllInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;
IWPAttrInterface Attr = Memo.TextAttr;
bool ok = false;

int cp = TextCursor.MarkerDrop(0);
// Use FindText.- If "MoveCursor=false", the text will be selected
TextCursor.FindText("some text", true, false, false, false, ref ok);
if (ok)
{
    Attr.Clear();
    Attr.SetFontface("Arial");
}
TextCursor.MarkerGoto(true, cp);
```

Instead, it has to be done like this:

```
IWPMemo Memo = wpdllInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;
bool ok = false;

int cp = TextCursor.MarkerDrop(0);
// Use FindText.- If "MoveCursor=false", the text will be selected
TextCursor.FindText("some text", true, false, false, false, ref ok);
if (ok)
{
    IWPAttrInterface Attr = Memo.TextAttr;
    Attr.Clear();
    Attr.SetFontface("Arial");
}
TextCursor.MarkerGoto(true, cp);
```

Or You use CurrSelAttr instead

```
IWPAttrInterface Attr = Memo.CurrSelAttr;
```

4.1.3.1 C) CurrAttr

Declaration
`IWPAttrInterface CurrAttr;`

Description
This is the interface to the **current writing attribute** of the active editor.

You can use it to change character attributes, such as the font name.

To change paragraph attributes please use CurrPar.

Please also see TextAttr and the topic "Font Attributes".

4.1.3.1 D) CurrSelAttr

Declaration
```
IWPAttrInterface CurrSelAttr;
```

Description
This interface can be used to change the attributes of the text which is currently selected.

Also see TextAttr!

4.1.3.1 E) CurrParAttr

Declaration
```
IWPAttrInterface CurrParAttr;
```

Description
This is the interface to manipulate the **text attributes** of the text in the current paragraph (the paragraph the cursor is located within).

You can use this interface to change the font of **all characters in the paragraph**. If you only need to change the attribute of certain characters in the paragraph use either CharAttr or SetCharAttr.

Example: The current paragraph should be bold and use the font "Verdana" with 10 pt.

```
IWPAttrInterface parattr = Memo.CurrParAttr;
parattr.IncludeStyles(1);
parattr.SetFontface("Verdana");
parattr.SetFontSize(10);
```

Do not use this interface to modify the attributes of the paragraph itself, such as the indent or also the default font for a paragraph.

4.1.3.1 F) CurrPar

Declaration
```
IWPParInterface CurrPar;
```

Description

This is the interface to manipulate the current paragraph. This is the paragraph the cursor (insertion marker) is located within.

When currently text is selected, CurrPar can be used to modify the attributes of the selected text.

You can use this interface to read the text, to manipulate the text and to change the attributes of the paragraph. To change the attributes of all characters in this paragraph use the interface provided by CurrParAttr. If you only need to change the attribute of certain characters use either CharAttr or SetCharAttr.

Example:


```
IWPMemo Memo = wpdllInt1.Memo;
IWPParInterface par = Memo.CurrPar;

// Clears the paragraph
par.SetText("",0);
// Appends new text using current writing mode
par.AppendText("Hello World",-1);
// activates all borders
par.Borders = 15;
// and shading
par.ParShading = 30;
par.ParColor = wpdllInt1.ToRGB(Color.Blue);

// read the character at the current position
par.GetChar(Memo.TextCursor.CPPosInPar)

// Change the indents
par.ParASet((int)WPAT.IndentLeft, 360);
par.ParASet((int)WPAT.IndentFirst, -360);
```

Important VB6 note:

When using certain interfaces of TextDynamic please first create a variable and assign the value to it. Otherwise You likely get the error "Block variable not defined".

```
Dim currpar As IWPParInterface
Set currpar = wpdllInt1.currpar
```

4.1.3.1 G) DefaultAttr

Declaration

```
IWPAttrInterface DefaultAttr;
```

Description

This is the interface to the default attribute of the active editor. The default attribute is used for text which does not have an attribute - this is the case for text with the CharAttrIndex 0.

```
IWPMemo Memo = wpdllInt1.Memo;
Memo.CurrPar.ClearCharAttr();
Memo.DefaultAttr.SetFontface("Courier New");
```

To define a font + size which is used for each new documents please use Command ID 18 - Set default font and size

4.1.3.1 H) CurrBand

Declaration

```
WPReportBand CurrBand;
```

Description
Reserved

4.1.3.1 I) CurrentZooming

Declaration
```
int CurrentZooming;
```

Description
This is the current zooming value. It is automatically adjusted when auto zooming is enabled.

4.1.3.1 J) CurrGroup

Declaration
```
IWPReportBand CurrGroup;
```

Description
Reserved

4.1.3.1 K) CurrObj

Declaration
```
IWPTextObj CurrObj;
```

Description
This is the interface to the object interface of the current object. The current object is the object which has been inserted last. Since the mail merge facility also uses text objects, you can also use CurrObj to manipulate fields created by InputField

This C# code inserts a horizontal line and then changes it's color to red.

```
wpdllInt1.TextCursor.InputObject(TextObjTypes.wpobjHorizontalLine,"","",0);
IWPTextObj obj = wpdllInt1.CurrObj;
if(obj!=null)
 obj.IntParam = wpdllInt1.ToRGB(Color.Red);
```

This VB code inserts a field and then makes it "editable"

```
Dim Cursor As IWPTextCursor
Dim Field As IWPTextObj
Set Cursor = WPDLLInt1.Memo.TextCursor
Set Field = WPDLLInt1.Memo.CurrObj

Cursor.InputText "Some text"
Cursor.InputField "FieldName", "Display Text", False
Field.Mode = 2 ' Make it Editable!
Cursor.InputText "Some more text"

'Switch to edit mode
WPDLLInt1.Memo.FormCompletion = True
```

4.1.3.1 L) CurrSelObj

Declaration

```
IWPTextObj CurrSelObj;
```

Description
This interface will be null unless an object is selected. If the property is != null you can use it to manipulate and examine the selected object.

Load new image data into the currently selected object:

```
IWPTextObj selobj =  wpdllInt1.CurrSellObj;
if ((selobj!=null)&&
    (selobj.ObjType==TextObjTypes.wpobjImage))
{
      selobj.LoadFromFile("c:\\Test.bmp");
}
```

4.1.3.1 M) CurrStyle

Declaration
```
IWPParInterface CurrStyle;
```

Description
This property can be used to manipulate the current style.

The "current style" is the style which has been added last or which was located by function SelectStyle()

Please note that the same interface type is used for styles and for paragraphs. This has to do with the internal implementation of paragraphs which inherit all properties and methods of styles.

When you use a IWPParInterface to manipulate a style the methods which change the text will have no effect. But you can use the properties, i.e. IndentLeft to change the left indent defined by this style.

Of course it is possible to use WPCSS strings to read the properties of a paragraph and assign to a style and vice versa.

VB.NET example:

```
Memo.CurrStyle.Alignment = 1 '0=lLeft, 1=Center, 2=Right, 3=Justified
```

Please note that you need to execute ReformatAll(**true**, true) after you have changed a style.

4.1.3.1 N) CurrStyleAttr

Declaration
```
IWPAttrInterface CurrStyleAttr;
```

Description
This interface is used to change the character attributes defined by the current style. Also see CurrStyle and SelectStyle.

VB.NET example:

```
Memo.CurrStyleAttr.SetFontface("Courier New")
Memo.CurrStyleAttr.SetFontSize("12")
```

Notes:

- You need to execute ReformatAll(**true**, true) after you have changed a style.
- The character attributes defined by a style will be only used by text which does not override these attributes. When you insert new text you can clear all writing attributes using Memo.CurrAttr.Clear().

4.1.3.1 O) LabelDef

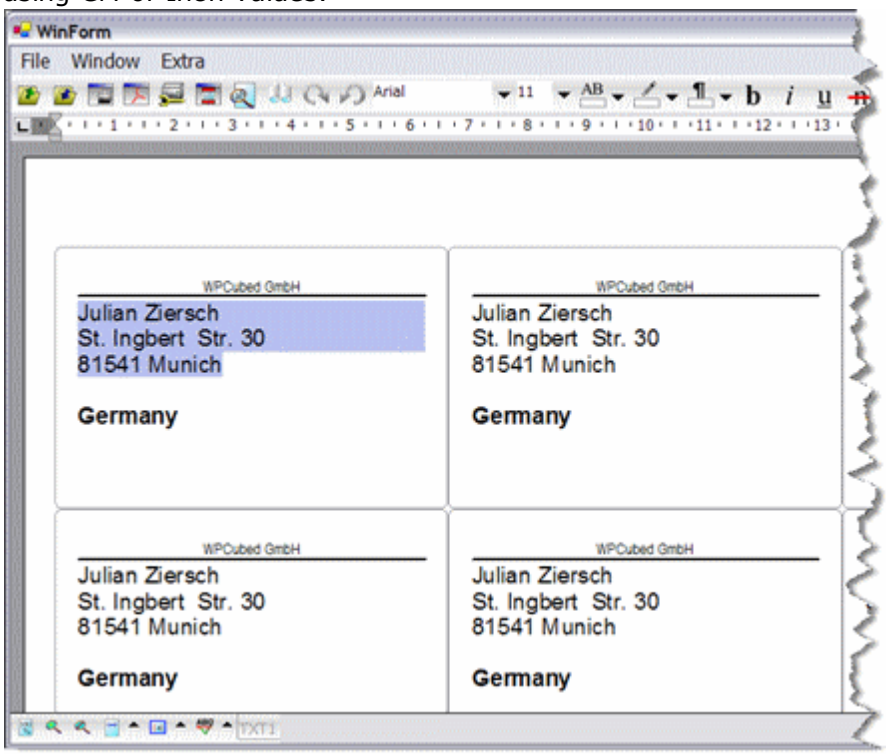
Interface to display labe sheet and change label setup

Declaration

```
IWPLabelDef LabelDef;
```

Description

Using the LabelDef interface you can quickly print labels. It is also possible to preview the label sheets just like they would be printed. It is even possible to edit the text on the label sheets. You can also specify the lablel number to start with. All parameters of a label can be specified, using CM or Inch values.



This C# code copies the current text (it expects this to be an address, about 5 lines) 30 times - each one on a new page. Then it activates the other text and activates the label display for this text.

```
private void LabelEdit_Click(object sender, System.EventArgs e)
{
 LabelEdit.Checked = !LabelEdit.Checked;
 if (LabelEdit.Checked)
 {
  // Delete the label text
  wpdllInt1.Memo.RTFDataDelete("LABELS");
  // Make 30 copies
  for (int i = 0; i < 30; i++)
  {
  wpdllInt1.Memo.RTFDataAppendTo("LABELS", true);
```

```
  }
  // and display the label sheet
  wpdllInt1.Memo.RTFDataSelect("LABELS");
  // and activates label display
  wpdllInt1.Memo.LabelDef.Caption = "WPCubed GmbH";
  wpdllInt1.Memo.LabelDef.Active = true;
 }
 else
 // switch back to normal text
 wpdllInt1.Memo.RTFDataSelect("@@FIRST@@");
}
```

Category
Mailmerge

4.1.3.1 P) LastFileName

Declaration
```
string LastFileName;
```

Description
The last file name which was used for file load or save.

4.1.3.1 Q) PageSize

Declaration
```
IWPPageSize PageSize;
```

Description
You can use this interface to modify the page size information stored with the document.

All properties are twips values. 1 twip = 1/1440 inch.

The method SetPageWH can be used to set multiple values. The value -1 means the current value is unchanged.

```
wpdllInt1.Memo.PageSize.SetPageWH(-1, -1, 720, 720, 720, 720);
```

To make the current page size the default for new documents use Memo.TextCommandStr (19,0,"");

```
wpdllInt1.Memo.TextCommandStr(19, 0, "");
```

4.1.3.1 R) PageSizeList

Declaration
```
IWPPageSizeList PageSizeList;
```

Description
The page size list allows it to provide a different page size (and margins) for each page in the document. Together with method GetPageAsMetafile this list makes it easy to display or print a document in a list of rectangular areas. The PageSizeList can be switched on and off using its property "Active". When it is active the page sizes defined in the document are overridden and the event OnMeasurePage is not been triggered.

Usually you have to execute ReformatAll to update the screen and layout information. (Not required by toggling "Active" or changing the Resolution.

4.1.3.1 S) PrintParameter

Declaration
```
IWPPrintParameter PrintParameter;
```

Description
This interface provides access to special printing options. You can print only the odd or even pages.

4.1.3.1 T) SelText

Declaration
```
string SelText;
```

Description
The property can be used to read the currently selected text as string. It does not save any special characters (such as object anchors) or formatting information. It works a lot faster than SaveToString since it does not have to use the text writer classes. You can also assign a value to this property to insert a string.
SelText can be used within the event OnCompleteWord to implement text macros.

Note: To insert RTF or HTML formatted strings instead of SelText use LoadFromString.

Also see
Memo.TextCommandStr ID 35/36/37 to change and read paragraphs, lines and words at cursor position.

4.1.3.1 U) Text

Get/Set text as unicode string

Declaration
```
string Text;
```

When Reading:
Retrieves the text in the editor as unicode string. Table cells will be delimited with TAB caracters, each paragraph and row will be closed with \r\n.
Use SaveToString to retrieve the text as RTF, ANSI, HTML or WPT string.

When Writing:
Load the text from a unicode string. Use LoadFromString to load the text as RTF, ANSI, HTML or WPT string. LoadFromString(string,true,"AUTO") will detect the format automatically.

This property works differently to Text and Text2 which always save formatting information and is able to load formatted text.

4.1.3.1 V) TextCursor

Methods and properties to insert text and change current position.

Declaration
```
IWPTextCursor TextCursor;
```

Description

This is the interface to the current cursor object.

The cursor interface contains most methods to create text and the properties which read and modify the current position in the text.

Please refer to `IWPTextCursor` for a list of methiods.

.NET: We recommend to assign the value of "TextCursor" to a variable and call wpdllint. ReleaseInt(variable) when the reference is not any longer required.

4.1.3.1 W) SpecialTextAttr

Declaration

```
IWPCharacterAttr SpecialTextAttr([In] SpecialTextSel Select);
```

Description

The method SpecialTextAttr() makes it possible to modify the appearance of hyperlinks, fields and other text. It provides you with a reference to the interface IWPCharacterAttr to change several properties. The parameter "Select" may have the following values:

```
0 : SpecialTextSel.wpHiddenText - hidden text
1 : SpecialTextSel.wpFootnote - footnote symbols
2 : SpecialTextSel.wpInsertpoints - the start and end markers of merge fields
3 : SpecialTextSel.wpHyperlink - the start and end marker of hyperlinks
4 : SpecialTextSel.wpSPANStyle - the start and end marker of inline SPAN styles
5 : SpecialTextSel.wpAutomaticText - merged text within fields
6 : SpecialTextSel.wpProtectedText - the text which has the protected property
7 : SpecialTextSel.wpBookmarkedText - the text within bookmark objects
8 : SpecialTextSel.wpInsertedText - not used
9 : SpecialTextSel.wpDeletedText - not used
10: SpecialTextSel.wpWordHighlight - highlighted words
11: SpecialTextSel.wpFieldTextObjects - single objects, such as page numbers.
```

Example:

```
WPDLLInt1.SpecialTextAttr(SpecialTextSel.wpInsertpoints).Hidden = true;
WPDLLInt1.SpecialTextAttr(SpecialTextSel.wpInsertpoints).CodeTextColor = 0;
```

4.1.3.2 Methods

4.1.3.2 A) AppendOtherText

Append the contents of second editor

Declaration

```
void AppendOtherText([In] int Mode);
```

Description

This procedure appends the text stored in the respective other editor (in case two editors are active in the control). You can use this methods with mail merge to create a long text with multiple letters.
To use AppendOtherText please initialize TextDynamic in the double editor mode, using SetEditorMode(1,...).

Alternatively you can use RTFDataAppendTo.

Parameter "Mode":

0: simply append the other text.

1: create a new page and appends the text.

2: create a new page and section and appends the text.

3: appends the text as new section (also copies header+footer!)

4: Only copy page size and header and footer

Category
Mailmerge

4.1.3.2 B) BlockAdd

Creates or selects a header or footer text layer

Declaration

```
IWPDataBlock BlockAdd([In] DataBlockKind Kind, [In] DataBlockRange Range, [In]
string Name, [In] int SectionID);
```

Description
This method is used to select an existing (with matching Range/Name) or create a new header or footer text block. An interface to the created block is returned as IWPDataBlock reference. This method works similar as TextCursor.InputHeader and TextCursor.InputFooter but will not modify the cursor position.

In the .NET assembly also available is
public ***bool BlockAddAndWorkOn****(DataBlockKind Kind, DataBlockRange Range, string Name, int SectionID)*
it creates the new block and places the cursor inside.

Note: Please don't forget to call ReleaseInt() with the returned interface at the end of your code.

Possible Values for DataBlockKind:
0 wpIsBody,
1 wpIsHeader,
2 wpIsFooter,
3 wpIsFootnote,
4 wpIsLoadedBody,
5 wpIsDeleted,
6 wpIsOwnerSelected

Possible Values for DataBlockRange:
0 wpraOnAllPages,
1 wpraOnOddPages,
2 wpraOnEvenPages,
3 wpraOnFirstPage,
4 wpraOnLastPage,
5 wpraNotOnFirstAndLastPages,
6 wpraNotOnLastPage,
7 wpraNamed,
8 wpraIgnored,
9 wpraNotOnFirstPage

The parameters Name and SectionID are usually not required. Please pass "" and 0.

VB.NET Example - create a header and move the cursor inside.

```
Dim header As IWPDataBlock
header = Memo.BlockAdd(DataBlockKind.wpIsHeader, DataBlockRange.wpraOnAllPages,
"", 0)
'Clears the text
header.Clear()
'Moves the cursor inside the header
header.WorkOnText = True

' it is required to call ReleaseInt to remove the interface
wpdllint.ReleaseInt(header)
```

This code will have the same effect: `Memo.TextCursor.InputHeader(DataBlockRange.wpraOnAllPages, "", "")`

C# Example - create a header and some text but do not move cursor:

```
IWPMemo Memo;
Memo = wpdllInt1.Memo;

IWPDataBlock header;
header = Memo.BlockAdd(DataBlockKind.wpIsHeader, DataBlockRange.wpraOnAllPages, "", 0);
//Clears the text
header.Clear();
header.AppendParagraph();
// change the default attribute for this paragraph
header.CurrParAttr.SetFontface("Courier New");
header.CurrParAttr.SetFontSize(22);
// Sets the text using the default attribute
header.CurrPar.SetText("Header text for the document", -1);
```

In case you need to append text or a text object (page numbering) using a different style you can use the AttrHelper!

```
// Get this character index
Memo.CurrAttr.CharAttrIndex = header.CurrParAttr.CharAttrIndex;
// and modify it
Memo.CurrAttr.IncludeStyles(1); // Bold!
// append text using the current writing mode
header.CurrPar.AppendText(", bold text", -2);
```

4.1.3.2 C) BlockAppend

Creates a new header or footer layer

Declaration

```
IWPDataBlock BlockAppend([In] DataBlockKind Kind);
```

Description

This method is used to create a new header or footer text block. An interface to the created block is returned as IWPDataBlock reference so you can modify the properties.

Note: Please don't forget to call ReleaseInt() with the returned interface at the end of your code.

```
Dim header As IWPDataBlock
```

```
header = Memo.BlockAdd(DataBlockKind.wpIsHeader, DataBlockRange.wpraOnAllPages,
"", 0)

....
' it is required to call ReleaseInt to remove the interface
wpdllint.ReleaseInt(header)
```

Hint: This action starts the "manage header and footer" dialog: **wpaDiaManageHeaderFooter**

4.1.3.2 D) BlockFind

Locates a header or footer layer.

Declaration

```
IWPDataBlock BlockFind([In] DataBlockKind Kind, [In] DataBlockRange Range, [In]
string Name, [In] int SectionID);
```

Description

This method is used to check if a certain header or footer layer exists. If it does, its ID, >0 is retuned. This ID can be used in even OnGetSpecialText to select this header or footer for a certain page.IWPDataBlock reference.

To delete a block (such a header or footer text) call the "Delete" method published by the interface IWPDataBlock.

Note: Please don't forget to call ReleaseInt() with the returned interface at the end of your code.

4.1.3.2 E) Clear

Clears the text buffer

Declaration

```
void Clear([In] bool KeepHeaderFooter, [In] bool KeepStyles);
```

Description

This method clears the text, optionally the header/footer texts and/or the styles can be kept.

Hint:

If you used memo.RTFDataSelect("SOME_NAME") to select a different RTF data block earlier in the program, you need to call memo.RTFDataSelect("@@FIRST@@") to select the default RTF data block to make sure the editor is in default state again.

4.1.3.2 F) CopyToClipboard

Copy text to clipboard

Declaration

```
void CopyToClipboard();
```

Category

Standard Editing Commands

4.1.3.2 G) CutToClipboard

Copy text to clipboard and delete it

Declaration

```
void CutToClipboard();
```

Category

Standard Editing Commands

4.1.3.2 H) PasteFromClipboard

Declaration
```
void PasteFromClipboard();
```

Description
Paste text from clipboard.

Category
Standard Editing Commands

4.1.3.2 I) DebugShowParProps

Display messagebox with paragraph properties

Declaration
```
void DebugShowParProps();
```

Description
This method can be used to check which attributes are used by the current paragraph.

TextDynamic:

A Message box will be displayed which displays the attributes and the attributes of ancestor paragraphs and styles.

RTF2PDF:

A debug message is sent which includes the attributes and the attributes of ancestor paragraphs and styles

4.1.3.2 J) DeleteLeadingSpace

Declaration
```
void DeleteLeadingSpace([In] bool EmptyFields, [In] bool InFirstPar);
```

Description
This procedure deletes all empty paragraphs at the beginning of the text. Also see methods see DeleteTrailingSpaces and DeleteParWithCondition.

Parameters	
InFirstPar	If true, the spaces at the beginning of first paragraph will be deleted.
EmptyFields	If true, paragraphs which are empty except for empty merge fields will be also deleted.

4.1.3.2 K) DeletePage

Deletes a certain logical page

Declaration
```
void DeletePage([In] int PageNr);
```

Description
The text has to be reformatted for this method to work. The procedure will try to delete the text which is displayed on the given page.

4.1.3.2 L) DeleteParWithCondition

Extensible function to delete paragraphs

Declaration

```
void DeleteParWithCondition([In] int Condition);
```

Description

This method has been reserved for intelligent paragraph deletion.
The following Conditions are supported:

0 : If a paragraph contains empty merge field(s) and no text, it is deleted. You can use it when doing mailmerge to erase empty lines.

New: You can **hide a paragraph temporarily** if you add the value 256 to the mode id.

If you also add 512 the previously already hidden paragraphs stay hidden.

Use the mode 257 to show all paragraphs again.

Category

Mailmerge

4.1.3.2 M) DeleteStyle

Declaration

```
void DeleteStyle([In] string StyleName);
```

Description

This method deletes the style with the given name.

Category

Paragraphstyle Support

4.1.3.2 N) DeleteTrailingSpaces

Declaration

```
void DeleteTrailingSpaces([In] bool EmptyFields);
```

Description

This method deletes spaces and empty paragraphs at the end of the text. Also see methods see DeleteLeadingSpace and DeleteParWithCondition.

Parameters	
EmptyFields	If true, paragraphs which are empty except for empty merge fields will be also deleted.

4.1.3.2 O) EnumDataBlocks

Declaration

```
void EnumDataBlocks([In] int EventParam);
```

Description

With EnumDataBlocks the event OnEnumDataBlocks will be called for each text layer in the document.

Parameters	
EventParam	This value will be passed through to the event handler.

Category
Callback Functions

4.1.3.2 P) EnumParagraphs

Declaration
```
void EnumParagraphs([In] bool OnlyActiveText, [In] int EventParam);
```

Description

The event OnEnumParOrStyle will be called for all paragraphs in the Document.

You can use TextCursor.CheckState(10) to force the PropChanged event which is required to update a data bound control.

Parameters	
OnlyActiveText	If true, only the paragraphs in the active text, this is the current text layer (such as the header or footer) or the text body, will be visited.
EventParam	This value will be passed through to the event handler.

Example:

Make all "bold" text "italic". This uses the method IWPParInterface.ParCommand.

The event handler:
```
private void wpdllInt1_OnEnumParOrStyle(object Sender, bool IsControlPar, int StartPos, i
{
                IWPParInterface Par = ParText;
                int a = 1;
                int p = 0;
                while(a>0)
                {
                        a = Par.ParCommand(7, (7 << 24) + 0, p);
                        if (a>=0)
                        {
                              Par.ParCommand(8, (7 << 24) + 1, a); // Adds "Italic"
                              Par.ParCommand(8, (6 << 24) + 0, a); // Removes "Bold"
                              p = (a >> 16) + (a & 0xFFFF); // Calculate Next position
                        }
                }
}
```

This code is used to start the process:
```
                IWPMemo Memo = wpdllInt1.Memo;
                Memo.EnumParagraphs(false, 1);
                wpdllInt1.Memo.ReformatAll();
```

Category
Callback Functions

4.1.3.2 Q) EnumParSiblings

Declaration
```
void EnumParSiblings([In] bool FromStart, [In] int EventParam);
```

Description
Enumerate all paragraphs in current paragraph nesting level and call event OnEnumParOrStyle. This are the siblings of the current paragraph. The function can be used if you need to check or modify all paragraphs in a cell.

Parameters	
FromStart	If true the methods starts with the first sibling, otherwise it will start with current paragraph.
EventParam	This value will be passed through to the event handler.

Category
Callback Functions

4.1.3.2 R) EnumParStyles

Declaration
```
void EnumParStyles([In] int EventParam);
```

Description
The event OnEnumParOrStyle will be called for all paragraph styles which are defined.

Parameters	
EventParam	This value will be passed through to the event handler.

To save the styles to a file use TextComandStr(28, filename) - to load the file use TextComandStr(29, filename)

Category
Callback Functions
Paragraphstyle Support

4.1.3.2 S) EnumSelParagraphs

Declaration
```
int EnumSelParagraphs([In] bool OnlyCompletePars, [In] bool ControlParsToo, [In]
int EventParam);
```

Description
The event OnEnumParOrStyle will be called for the selected paragraphs.

Parameters	
OnlyCompletePars	If true, only the paragraphs which are completely selected will trigger the event.
ControlParsToo	If true also table and table row paragraphs will trigger the event.
EventParam	This value will be passed through to the event handler.

Category
Callback Functions

4.1.3.2 T) EnumTextObj

Declaration
```
int EnumTextObj([In] TextObjTypes ObjType, [In] bool OnlyActiveText, [In] string
```

```
NameStartsWith, [In] int EventParam);
```

Description
This method allows it to check and modify all text objects in the current document. Such text objects are the tags for bookmarks, hyperlinks, mail merge fields and also embedded images and text object fields.

The event OnEnumTextObj is triggered for each objects which meets the specified criteria. This event received an IWPTextObj interface which let you modify the the object.

Parameters	
ObjType	You can set this parameter to 0 to report all objects or use the object type ids, 1=fields, 2=hyperlinks, 3=bookmarks, 7=text object, 8=page reference, 11=footnote, 12=image or text box and 13 for horizontal lines.
OnlyActiveText	If true only check objects in the active text (header/footer, text body)
NameStartsWith	Only objects which name starts with this string will be reported. This is useful to enumerate a group of merge fields, for example to select all fields for a certain database.
EventParam	This value will be passed through to the event handler.

Category
Callback Functions

4.1.3.2 U) SelectStyle

Declaration

```
void SelectStyle(string StyleName);
```

Description
Selects a certain style. The style can be then manipulated through interface CurrStyle and CurrStyleAttr.

VB.NET example:

```
'initialize the reference to "Memo"
Dim Memo As IWPEditor
Memo = Me.WpdllInt1.Memo
'Create a new header (for all pages) and place the cursor inside
Memo.TextCursor.InputHeader(0, "", "")
'create a text style called "myStyle", this is not used for the actual header,
confusing isn't it
'Selects a certain style. The style can be then manipulated through interface
CurrStyle.
Memo.SelectStyle("myStyle")
'This interface is used to change the character attributes defined by the
current style.
Memo.CurrStyleAttr.SetFontface("Times New Roman")
Memo.CurrStyleAttr.SetFontSize("32")
'Now set paragraph attributes
Memo.CurrStyle.Alignment = 1 '0=lLeft, 1=Center, 2=Right, 3=Justified
'now apply this style to the current paragraph
Memo.CurrAttr.Clear()
```

```
'clear the writing attributes otherwise the style is not used
'because the style attributes are overridden by the character attributes
Memo.CurrPar.StyleName = "myStyle"
Memo.TextCursor.InputText("Header added programmatically") ' can't see how to
add page numbers etc
'Goto Body
Memo.TextCursor.GotoBody()
'This is required if you change a paragraph style
Memo.ReformatAll(True, True)
```

Please also see InputRowStart - there we added an VB.NET example which creates a table using merged cells and paragraph styles.

Category
Paragraphstyle Support

4.1.3.2 V) LoadNumberStyles

Declaration
```
function LoadNumberStyles(const Filename: WideString; Mode: Integer): Integer
```

Description

Loads the numberstyles from a file.

If the second parameter is 1, the styles will be merged.

This method should be only called with an empty editor!

4.1.3.2 W) SaveNumberStyles

Declaration
```
procedure SaveNumberStyles(const Filename: WideString)
```

Description

Saves the number styles into a file.

4.1.3.2 X) LoadStyleSheet

Declaration
```
function LoadStyleSheet(const filename: WideString): Integer;
```

Description

Loads the paragraph styles from a file.

In case the filename has the extension .css the CSS Level 2 syntax is used.

Otherwise the proprietary TextDynamic/WPTools style syntax is used.

If you need to load CSS data from a string please use TextCommandStr(30, "...")

4.1.3.2 Y) SaveStyleSheet

Declaration
```
procedure SaveStyleSheet(const filename: WideString);
```

Description

Saves the paragraph styles to a file.

In case the filename has the extension .css the CSS Level 2 syntax is used.

Otherwise the proprietary TextDynamic/WPTools style syntax is used.

4.1.3.2 Z) FindFooter

Declaration

```
int FindFooter([In] int Range, [In] string Name);
```

Description

Locates a footer text for a certain "range" and returns its ID. If the text block was not found 0 is returned.

Category

Hader and Footer Support

4.1.3.2 AA) FindHeader

Declaration

```
int FindHeader([In] int Range, [In] string Name);
```

Description

Locates a header text for a certain "range" and returns its ID. If the text block was not found 0 is returned.

Category

Hader and Footer Support

4.1.3.2 AB) GetNumberStyle

Declaration

```
IWPNumberStyle IWPNumberStyle GetNumberStyle([In] int ID, [In] int Mode, [In]
int Level);
```

Description

This method can be used to modify number styles. Usually the parameter ID will be passed as value 0.

In case you want to add a new style set, pass ID as -1.

Parameters

ID: If the ID is >0 a numberstyle will be selected by ID (WPAT_NumberStyle) and level (WPAT_NumberLevel).
If Mode = -1000 the numberstyle (if one was found) will be deleted. If no style was found or the style was deleted the result value is null.
If ID=-1 a new numbering style will be created. This style can be either a simple numbering style (Level=0) or an outline (Level between 1 and 9)

Mode: The numbering mode, possible values are:
0 : no numbering
1 : bullets
2 : circles (not used)
3 : arabic numbering 1,2,3
4 : captital roman
5 : roman
6 : capital latin
7 : latin

Special Mode: If ID>0 the value -1000 will cause the deletion of this style.

Level — This is the outline level between 1 and 9
0 selects simple numbering (not nestable)

Returns
IWPNumberStyle reference or null if undefined.

You can also use it to modify the outline (level<>0)

Example:
```
IWPNumberStyle NStyle = rtF2PDF1.Memo.GetNumberStyle(0, 0, 1);
NStyle.ASet((int)WPAT.NumberMODE,1); // Make it Arabic 1. 2. 3. ...
rtF2PDF1.ReleaseInt(NStyle);
```

The value "ID" can be used with the property **WPAT_NumberStyle** in any paragraph.

```
Example - create custom bullets

            IWPMemo Memo = wpdllInt1.Memo;
            IWPAttrInterface Attr = Memo.TextAttr;
            IWPNumberStyle style;
            style = Memo.GetNumberStyle(-1,1,0); // Bullet

            if(style!=null)
            {
                style.TextA = "¢"; // after text = "."
                style.Font="WingDings";
                style.Indent = 360;
            }
            Attr.AttrSET((int)WPAT.NumberSTYLE, style.ID);

            wpdllInt1.ReleaseInt(style);
```

Note: Please don't forget to call ReleaseInt() with the returned interface at the end of your code.

4.1.3.2 AC) GetObjAtXY

This method searches for an object at a certain x,y position.

Declaration
```
IWPTextObj IWPTextObj GetObjAtXY([In] int X, [In] int Y, [In] int TypeMask, [In]
int Mode);
```

Description
The Result is a reference to the IWPTextObj interface. If no object was found the result will be null.

This code locates the hyperlink at the mouse position:

```
private void wpdllInt1_MouseDown(object sender, System.Windows.Forms.MouseEventA
{
   IWPMemo Memo = wpdllInt1.CurrMemo;
   IWPTextObj obj = Memo.GetObjAtXY(e.X, e.Y, 0, 2);
}
```

Parameters	
X	The horizontal position
Y	The vertical position
TypeMask	Selects the object types to find: 0=automatic, find nearest object or the object at this position. 1..13=values as defined by enum "TextObjTypes". ie: 1=merge field, 2=hyperlink, 3=bookmark, 7=text object such as page number, 12=image object.
Mode	Selects the way the X and Y parameter is expected: -1: Dont use X and Y parameter. Use current mouse position instead. 0 : X and Y are client coordinates of the editor. If you use the method in OnMouseDown or OnMouseMove use this mode. 1 : X and Y are coordinates relative to upper left corner of edit control (including the toolbars) 2 : X and Y are screen coordinates (relative to desktop)

Category
Hyperlinks and Bookmarks

4.1.3.2 AD) GetPosAtXY

This method searches for the character position at x,y.

Declaration

```
bool GetPosAtXY([In] int X, [In] int Y, [In] int Mode, out int CPPos);
```

Description
The Result is false if no text was found.

This code inserts the character X at the click position:

```
private void wpdllInt1_MouseDown(object sender, System.Windows.Forms.MouseEventArgs e)
{
   IWPMemo Memo = wpdllInt1.CurrMemo;
   int pos;
   if (Memo.GetPosAtXY(e.X,e.Y,0,out pos))
   {
    int oldpos = Memo.TextCursor.CPPosition;
    Memo.TextCursor.CPPosition = pos;
    Memo.TextCursor.InputString("X",0);
    Memo.TextCursor.CPPosition = oldpos;
    Memo.ReformatAll(false,true);
  }
 }
```

Parameters	
X	The horizontal position
Y	The vertical position
Mode	Selects the way the X and Y parameter is expected: 3 or -1: Don't use X and Y parameter. Use current mouse position instead. 0 : X and Y are client coordinates of the editor. If you use the method in OnMouseDown or OnMouseMove use this mode. 1 : X and Y are coordinates relative to upper left corner of edit control (including the toolbars) 2 : X and Y are screen coordinates (relative to desktop) If bit 8 (value=256) is set in "Mode" the CurrPar interface will be updated to work with the found paragraph. In this case Pos will be the offset in this paragraph.

out CPPos	The absolute character position.

For .NET Drag and Drop set the property **AllowDrop** to true.

Then you need two event handlers:

```
private void wpdllInt1_DragDrop(object sender, System.Windows.Forms.DragEventArgs e)
{
     wpdllInt1.TextCursor.InputText("Dropped text");
     wpdllInt1.Reformat();
}

private void wpdllInt1_DragOver(object sender, System.Windows.Forms.DragEventArgs e)
{
   int pos;
   if (wpdllInt1.Memo.GetPosAtXY(0,0,-1, out pos))
   {
                wpdllInt1.Memo.TextCursor.CPPosition= pos;
         e.Effect=DragDropEffects.All;
   }  else e.Effect=DragDropEffects.None;
}
```

4.1.3.2 AE) GetRTFVariable

Declaration
```
string GetRTFVariable([In] string Name);
```

Description
Retrieves the value of a certain string variable which was stored with the document.

Category
Document Properties

4.1.3.2 AF) GetXY

Retrieves different X,Y positions from the editor

Declaration
```
void GetXY([In] int Mode, [In, Out] ref int X, [In, Out] ref int Y);
```

Description
You can use this method to receive different coordinate values. Mode selects the property to read:
0: X and Y are the position of the cursor in editor X,Y coordinates.
1: X and Y are the position of the cursor in client X,Y coordinates.
2: X and Y are the position of the cursor in screen X,Y coordinates.
3: X and Y are the position of the cursor on the page in twips (inch /1440).
4 : X and Y are the baseline of the current paragraph in client coordinates.
5 : X and Y are the baseline of the current paragraph in screen coordinates.
6 : X and Y are the upper left corner of the selected text in screen coordinates.
7 : X and Y are the lower right corner of the selected text in screen coordinates.
8 : X and Y are the upper left corner of editor in screen coordinates.
9 : X and Y are the lower right corner of the editor in screen coordinates.
10: Get horizontal and vertical scroll position.
11: **Set** horizontal and vertical scroll position. Use value=-1 to not set the X or Y scrolling.
12: X and Y are the width and height of the virtual desktop inside the editor.
13: X and Y are the width and height of the text.
14: X and Y are physical margin used by the current printer. (PrintXOffset, PrintYOffset)

15: X and Y are the current logical horizontal and vertical resolution used by the rendering engine. Usually this value is 600. (CurrentXPixelsPerInch, CurrentYPixelsPerInch)
16: X and Y are the current mouse cursor position.

Example VB6:

```
    Dim x As Long
    Dim y As Long
    Memo.GetXY 2, x, y
```

Category
Coordinate Conversion

4.1.3.2 AG) LoadFromFile

Declaration

```
bool LoadFromFile([In] string filename, [In] bool Insert, [In] string FormatStr
);
```

Description
Load a file into this editor.

Parameters	
filename	The name of the file
Insert	true to insert the file at the cursor position, otherwise the complete text is replaced.
FormatStr	Select the reader and set options. It is also possible to just the contents of a merge field using the syntax f:name=

This VB.NET example code display a file open dialog and loads the selected file into the editor.

```
    Dim OpenFileDialog As New OpenFileDialog
    OpenFileDialog.InitialDirectory = My.Computer.FileSystem.SpecialDirectories.
MyDocuments
    OpenFileDialog.Filter = "Textfiles (*.rtf,*.wpt,*.htm,*.html)|*.rtf;*.wpt;*.
htm;*.html|All Files (*.*)|*.*"
    If (OpenFileDialog.ShowDialog(Me) = System.Windows.Forms.DialogResult.OK)
Then
            Dim FileName As String = OpenFileDialog.FileName
            Dim memo As WPDynamic.IWPMemo
            memo = WpdllInt1.Memo
            If (memo Is Nothing) Or _
               Not memo.LoadFromFile(FileName, False, "AUTO") Then
                MessageBox.Show("Cannot load " + FileName)
            End If
    End If
```

Tip: In ASP projects you can use Server.MapPath(".") to get the current directory.

```
   if (!wpdllint1.Memo.LoadFromFile(Server.MapPath(".") + "test.rtf",true,""))
      wpdllint1.TextCursor.InputText("Cannot open file TEST.RTF");
```

Tip: In case the addon wphttpget.dll was activated using Memo.TextCommandStr(6,1, dll_path)
LoadFromFile can be also used with a http path to load data from a web server.

Category
Load and Save

4.1.3.2 AH) LoadFromStream

Declaration
```
bool LoadFromStream([In] object Stream, [In] bool Insert, [In] string FormatStr
);
```

Description
Load data from a stream into this editor.

.NET: The stream converter must be re-created for each load and save operation!

Example:
```
System.IO.Stream str = new System.IO.MemoryStream();
wpdllint1.Memo.SaveToStream( new WPDynamic.Stream2WPStream(str) ,false,"RTF");
wpdllint1.Position = 0;
wpdllint2.Memo.LoadFromStream(new WPDynamic.Stream2WPStream(str), true, "AUTO");
```

Parameters	
Stream	This can be either a reference to a IStream interface or a a reference to the Stream2WPStream converter provided by the .NET assembly.
Insert	true to insert the file at the cursor position, otherwise the complete text is replaced.
FormatStr	Select the reader and set options. see FormatStrings. It is also possible to just the contents of a merge field using the syntax f:name=

Category
Load and Save

4.1.3.2 AI) LoadFromString

Declaration
```
bool LoadFromString([In] string Data, [In] bool Insert, [In] string FormatStr);
```

Description
Load data into this editor.

Parameters
Data - The text to be "loaded".

Insert - true to insert the file at the cursor position, otherwise the complete text is replaced.

FormatStr - Select the reader and set options. See FormatStrings.
It is also possible to just the contents of a merge field using the syntax f:name=

Tip: In many cases using LoadFromVar will be more efficient:
```
 editorToPreview.Memo.LoadFromVar(
    editorToBePreviewed.Memo.SaveToVar(false, "RTF"),
    false, "RTF");
```

Category
Load and Save

4.1.3.2 AJ) LoadFromVar

Loads the data from a variant variable

Applies to
IWPMemo

Declaration

```
bool LoadFromVar([In, MarshalAs(UnmanagedType.Struct)] object Data, [In] bool
Insert, [In] string FormatStr);
```

Description

This method loads or inserts the contents of a variant - usually a variant array of bytes. You can use this method to load the data saved by SaveToVar.

Note: variant array of bytes work much more efficient to store RTF data than strings.

C# Example - copy text from one editor to another using SaveToVar/LoadFromVar

```
object currtext = wpdllInt1.Memo.SaveToVar(false, "WPT");
wpdllInt2.Memo.LoadFromVar(currtext, false, "AUTO");
```

Parameters

Data - The text to be loaded or inserted, variant or "object".
Insert - true to insert the text at the cursor position, otherwise the complete text is replaced.
FormatStr - Select the reader and set options. See FormatStrings.

Category

Load and Save

4.1.3.2 AK) SaveToFile

Declaration

```
bool SaveToFile([In] string filename, [In] bool OnlySelection, [In] string
FormatStr);
```

Description

Save text to a file. The engine will determine the format using the extension - unless you specify it using a format string such as "RTF", "HTML", "ANSI", "UNICODE" or "MIME".

Parameters	
filename	The name of the file
SelectionOnly	true to only save the selected text.
FormatStr	Select the writer and set options. See FormatStrings.

This VB.NET example code display a file save dialog and saves the text in the editor:

```
    Dim SaveFileDialog As New SaveFileDialog
    SaveFileDialog.InitialDirectory = My.Computer.FileSystem.SpecialDirectories.
MyDocuments
    SaveFileDialog.Filter = "Textfiles (*.rtf,*.wpt,*.htm,*.html)|*.rtf;*.wpt;*.
htm;*.html|All Files (*.*)|*.*"

    If (SaveFileDialog.ShowDialog(Me) = System.Windows.Forms.DialogResult.OK)
Then
            Dim FileName As String = SaveFileDialog.FileName
            If Not WpdllInt1 Is Nothing Then
                Dim memo As WPDynamic.IWPMemo
                memo = WpdllInt1.Memo
                If Not memo.SaveToFile(FileName, False, "AUTO") Then
                    MessageBox.Show("Cannot write " + FileName)
                End If
```

```
            End If
    End If
```

If you want to send an HTML e-mail from Outlook You can use this VB code:

```
  Dim olkApp As Outlook.Application, _
        olkMsg As Outlook.MailItem, _
        objFSO As FileSystemObject, _
        objFile As TextStream

    'Save File as HTML
    Dim Result As Boolean
    Dim sFilename As String

    sFilename = "C:\xxxxx.htm"
    Result = Me.WPDLLInt0.memo.SaveToFile(sFilename, False, "HTML")

    'Read the document in using the FileSystemObject
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    Set objFile = objFSO.OpenTextFile(sFilename, 1, False)
    'Get the open instance of Outlook
    Set olkApp = GetObject(, "Outlook.Application")
    'Create a new message
    Set olkMsg = olkApp.CreateItem(0)
    'Read the saved Word doc into the body of the message
    olkMsg.HTMLBody = objFile.ReadAll
    'Display the message
    olkMsg.Display
    'Destroy all the objects to avoid memory leaks
    Set olkMsg = Nothing
    objFile.Close
    Set objFile = Nothing
    Set objFSO = Nothing
```

Category
Load and Save

4.1.3.2 AL) SaveToStream

Declaration

```
bool SaveToStream([In] object Stream, [In] bool OnlySelection, [In] string
FormatStr);
```

Description
Save data to a stream.

.NET: The stream converter `Stream2WPStream` must be re-created for each load and save operation!

Example:

```
System.IO.Stream str = new System.IO.MemoryStream();
Memo.SaveToStream( new WPDynamic.Stream2WPStream(str) ,false,"RTF");
str.Position = 0;
Memo.LoadFromStream(new WPDynamic.Stream2WPStream(str), true, "AUTO");
```

Tip: You can use SaveToStream to load into an **.NET** array of bytes:

```
wpdllint1.Memo.SaveToStream(new WPDynamic.Stream2WPStream(stream), false, "RTF"
byte[] buf = new byte[stream.Length];
stream.Seek(0,0); // don't forget!
stream.Read(buf,0,buf.Length);
```

This code has the same effect as

```
object buf = wpdllint1.Memo.SaveToVar(false, "RTF");
if(buf!=null) Response.BinaryWrite((byte[])buf);
```

Parameters:
Stream : This can be either a reference to a IStream interface or a a reference to the **Stream2WPStream** converter provided by the .NET assembly.
SelectionOnly : true to only save the selected text.
FormatStr : Select the writer and set options. See FormatStrings.

Category
Load and Save

4.1.3.2 AM) SaveToString

Declaration

```
string SaveToString([In] bool OnlySelection, [In] string FormatStr);
```

Description
Save text to a string. To create a HTML string use the format string "HTML". If the embedded images should be saved to a certain directory "path" you can specify it using the format string HTML-imgpath:\"path\".

This method creates a unicode string which uses more memory than actually required (RTF and HTML use only single bytes). You can use SaveToVar to create a variant array of bytes.

Tip: To create a **.NET** array of bytes use SaveToVar or SaveToStream.

Tip: You can create a MIME encoded multipart e-mail data using the format string "MIME". (Please see chapter "Create MIME encoded e-mail data" in the manual)

Parameters:
OnlySelection : true to save the text which is selected.
FormatStr : Select the writer and set options. See FormatStrings.

Category
Load and Save

4.1.3.2 AN) SaveToVar

Saves to a variant variable

Declaration

```
object SaveToVar([In] bool OnlySelection, [In, MarshalAs(UnmanagedType.BStr)]
string FormatStr);
```

Description
This method saves the contents or the selected text to a variant array of bytes. This method requires less resources than SaveToString.

It can be

C# Example - copy text from one editor to another using SaveToVar/LoadFromVar

```
object currtext = wpdllInt1.Memo.SaveToVar(false, "WPT");
wpdllInt2.Memo.LoadFromVar(currtext, false, "AUTO");
```

Parameters

SelectionOnly - true to only save the selected text.

FormatStr - Select the writer and set options. See FormatStrings.

Returns
A variant or "object". The created data is null or an array of bytes!

ASP.NET example:

```
Response.Clear();
// Add new header  for RTF
Response.ContentType = "application/rtf";
Response.AddHeader("Content-Type", "application/rtf");
Response.AddHeader("Content-Disposition","inline;filename=" + afile + ".rtf");
object buf = wpdllint1.Memo.SaveToVar(false, "RTF");
if(buf!=null) Response.BinaryWrite((byte[])buf);
```

Category
Load and Save

4.1.3.2 AO) MergeText

Start the data merging process fo all or a group of fields.

Applies to
IWPMemo

Declaration

```
void MergeText([In] string FieldName);
```

Description
Start the data merging process. The event OnFieldGetText will be triggered for each mail merge field.

Mail merge fields can be inserted using IWPTextCursor.InputField.

It is also possible to use bookmarks and hyperlinks as merge fields. **This makes it possible to perform nested merge operations.**

To do so, please execute Memo.TextCommand(26, 2, 0) to merge in hyperlinks, Memo.
TextCommand(26, 3, 0) to merge into bookmarks
and Memo.TextCommand(26, 1, 0) to use merge fields (default).

Note:
TextDynamic uses standard RTF tags to load and save fields so merge fields inserted in Word are usually available in TextDynamic as well (without the extended attributes). Alternatively you can inserted the tags using escape text, i.e. <name> and use the method IWPTextCursor.FieldsFromTokens(StartText,EndText,FieldPreText) to convert the text into merge fields.

Parameter "Name":
This is the name of the fields to be merged. You can use the wildcard * to process a group of fields, i.e. MergeText("CustomerDB.*").

Please also see Category Mailmerge and the "Mail merge API introduction".

An alternative to do mail merge uses the method TextCursor.CPMoveNextObject

Here you can specify 1 for ObjType to find the merge fields.

You can then use CurrObj to update the data according to CurrObj.Name.

4.1.3.2 AP) Print

Declaration
```
void Print();
```

Description
Print the text. Optionally use BeginPrint/EndPrint to print several texts into one printing cue.

Note: When using Visual Basic 6, instead of method "Print" please use PrintPages 1,999999.

Note: You can also use Memo.TextCommandStr(11, range). to start printing. Here you can select odd or even pages and also pass a pages list.

New: The standard version of the product RTF2PDF / TextDynamic Server now also prints (but not threadsavely).

Category
Printing

4.1.3.2 AQ) PrintPages

Declaration
```
void PrintPages([In] int FromPage, [In] int ToPage);
```

Description
Print a range of pages. Optionally use BeginPrint/EndPrint to print several texts into one printing cue.

New: The standard version of the product RTF2PDF / TextDynamic Server now also prints (but not threadsavely).

Category
Printing

4.1.3.2 AR) PtrCommand

Declaration
```
int PtrCommand([In] int PtrComID, [In] int PtrValue, [In] int param, [In] string StrParam);
```

Description
Reserved for custom enhancements.

4.1.3.2 AS) Reformat

Declaration
```
void Reformat();
```

Description
Formats the text the next time the application is idle. Formatting is required after the creation of text, for example using TextCursor.InputString.

Please also see TextCommand ID 18, here we added options to force reformat and spellcheck.

4.1.3.2 AT) ReformatAll

Declaration
```
void ReformatAll([In] bool InitAll, [In] bool Repaint);
```

Description
Formats the text at once.

Optionally initializes (= calculate the font heights) and paints it.

Formatting is required after the creation of text, for example using TextCursor.InputString.

Initialization is only required after paragraph styles have been updated, otherwise the engine knows which paragraphs have to be initialized.

The formatting is performed at once, not in the next idle phase. If called in a loop, this can slow down the program significantly!

Therefore please also see **TextCommand(18, ..)** which is used to start the reformat process in the next idle phase. Especially when used inside of events, this would be the better way to initialize the text.

4.1.3.2 AU) RTFDataAdd

Declaration
```
void RTFDataAdd([In] string Name, [In] string Caption);
```

Description
Add an additional document to the TextDynamic controller. This document can then be selected with RTFDataSelect. The current document is not deleted by this command.
You have to specify a unique identifier name and an optionla caption. The caption will be used as caption for buttons in the tab set "Panel H2" - so it should be short. If the caption is empty the button will not be created.

Please see example "SimulatedMDI" in the manual.

Parameters	
Name	Unique ID
Caption	Short Caption

Category
Logical MDI Support

4.1.3.2 AV) RTFDataAppendTo

Declaration
```
void RTFDataAppendTo([In, MarshalAs(UnmanagedType.BStr)] string Name, [In] int
Options);
```

Description
This method can be used to append the current text to an existing or new text buffer. The name of the buffer is specified as parameter one.

When you are done with the merging you can switch to the result buffer using RTFDataSelect.

This C# code copies the current text (it expects this to be an address, about 5 lines) 30 times - each one on a new page. Then it activates the other text and activates the label display for this text.

```
private void LabelEdit_Click(object sender, System.EventArgs e)
{
    LabelEdit.Checked = !LabelEdit.Checked;
    if (LabelEdit.Checked)
    {
        // Delete the label text
        wpdllInt1.Memo.RTFDataDelete("LABELS");
        // Make 30 copies
        for (int i = 0; i < 30; i++)
        {
            wpdllInt1.Memo.RTFDataAppendTo("LABELS", 1);
        }
        // and display the label sheet
        wpdllInt1.Memo.RTFDataSelect("LABELS");
        // and activates label display
        wpdllInt1.Memo.LabelDef.Caption = "WPCubed GmbH";
        wpdllInt1.Memo.LabelDef.Active = true;
    }
    else
        // switch back to normal text
        wpdllInt1.Memo.RTFDataSelect("@@FIRST@@");
}
```

Please avoid to append the text to the source element (itself) - this causes the text to grow exponentially.

Parameters:

Name	The name of the destination RTF data element. Use any name or @@FIRST@@ for text in first editor and @@SECOND@@ for text in second editor (unless RTFDataSelect was used to display different element).
Options	This is a bit-field 1: Create a new page break before each copy 2: Don't copy merge fields 4: Don't copy bookmark tags 8: Don't copy images 64: Create a new section 128: assign page format of the source to the section 256: set the Reset-Outline-Numbers flag for the new section

Category
Logical MDI Support
Mailmerge

4.1.3.2 AW) RTFDataDelete

Declaration

```
void RTFDataDelete([In, MarshalAs(UnmanagedType.BStr)] string Name);
```

Description
Deletes a document with a given name.

Please make sure you select another, valid document before the deletion.

We recommend to not delete the last RTFData object but to clear it.

Category
Logical MDI Support
Logical MDI Support

4.1.3.2 AX) RTFDataSelect

Declaration
```
bool RTFDataSelect([In] string Name);
```

Description
Selects a document stored by the TextDynamic controller.

The name of the standard document used be editor #1 is "@@FIRST@@", for editor #2 it is "@@SECOND@@".
But it is possible to add additional documents using RTFDataAdd.

Example: See "Create mailing labels"

Category
Logical MDI Support

4.1.3.2 AY) GetPageAsMetafile

Declaration
```
int GetPageAsMetafile(int PageNr,int Options);
```

Description
Creates a meta file from a certain page and returns the handle. Please call Memo.ReformatAll (false,false) if the editor was dynamically created.

To create a file use SavePageAsMetafile.

Also see the commands to create bitmap files (BMP, PNG and TIFF).

Parameters:
PageNr : The number of the page to be saved. The first is 0.
Options: a bit field:
4: display a frame for the page margins
8: optimized for PDF export. We recommend to always set this bit!
16: also print selection marker
32: do not print watermarks
64: do not print header and footer
128: do not print images
256: print table grid lines.
512: Export embedded meta-files as bitmaps
1024: Create a metafile with current screen resolution, otherwise the resolution is 600

Returns
The handle to an enhanced metafile. If an error happens the return value is 0. The handle must be freed by the caller.

This C# function can be used to load a metafile from a handle into picture box:

```
private void LoadInImage(System.Windows.Forms.PictureBox PictureBox, int MetaHandle)
{
 if (MetaHandle!=0)
 {
  Metafile aMetafile=new Metafile(new IntPtr(MetaHandle), true);
  PictureBox.Image = aMetafile;
  PictureBox.SizeMode =  PictureBoxSizeMode.StretchImage;
 }
 else PictureBox.Image = null;
}
```

You can use it in code like this:

```
LoadInImage(pictureBox1,
 wpdllInt1.Memo.GetPageAsMetafile(0,0));
```

Example using a dynamic editor:

```
   WPDLLInt temp_editor = new WPDLLInt();
   temp_editor.Memo.LoadFromString( data_provided_as_string , false, "AUTO");
   temp_editor.SpecialTextAttr(SpecialTextSel.wpInsertpoints).Hidden = true;
   temp_editor.Memo.ReformatAll(true, false);
   for(int i = 0; i < temp_editor.Memo.TextCursor.PageCount; i++)
   {
            IntPtr metaHandle = temp_editor.Memo.GetPageAsMetafile(i, 8);
            if (metaHandle.ToInt32() == 0)
            {
                     Console.WriteLine("page " + i.ToString() + " not loaded");
            }
            else
            {
                     System.Drawing.Imaging.Metafile aMetafile = new System.Drawing.Imagi
                     aMetafile.Save("c:\\page__" + i.ToString() + ".emf");
            }
   }
   temp_editor.DisposeEditor();
```

Category
Load and Save

4.1.3.2 AZ) SavePageAsMetafile

Declaration
```
bool SavePageAsMetafile(int PageNr, string : Filename, int Options, int
BackgroundColor);
```

Description
Creates a EMF file from a certain page. Also see method GetPageAsMetafile.

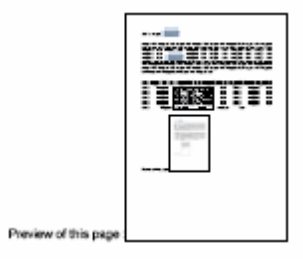
Also see the commands to create bitmap files (BMP, PNG and TIFF).

Parameters:
PageNr : The number of the page to be saved. The first is 0.

Filename: The enhanced metafile (*.EMF) file to be created.
Options: a bit field:
4: display a frame for the page margins
8: optimized for PDF export. We recommend to always set this bit!
16: also print selection marker
32: do not print watermarks
64: do not print header and footer
128: do not print images
256: print table grid lines.
512: Export embedded meta-files as bitmaps
BackgroundColor: A background color as RGB value

Example:

```
// Create a preview (thumbnail) of the
document INSIDE the
 // document. We use a temporary file.
 string tempfile = System.IO.Path.GetTem
 string emf_tempfile = tempfile + ".EMF"
 try
 {
     // Format the text, othewise SavePa
     Memo.ReformatAll(true,false);
     // save this page as image
     Memo.SavePageAsMetafile(0,emf_tempf
     // and insert it here
     TextCursor.InputText("Preview of th
     TextCursor.InputImage(emf_tempfile,
     Memo.CurrObj.Frame = 1;
     // Scale to 30 % of the original si
     Memo.CurrObj.ScaleSize(0,0,30);

     // The preview does not contain the
     // so recreate the image file and l
     Memo.ReformatAll(false,false);
     Memo.SavePageAsMetafile(0,emf_tempf
     Memo.CurrObj.Contents_LoadFromFile(
 }
 finally
 {
     // Delete the temporary files
     System.IO.File.Delete(emf_tempfile)
     System.IO.File.Delete(tempfile);
 }
```



Category
Load and Save

4.1.3.2 BA) GetPageAsBitmap

Declaration

```
int GetPageAsBitmap(
      int PageNr;
      string  filename;
      int Xres;
      int Yres;
```

```
    int Mode);
```

Description

This API controls the RTF to Bitmap conversion of TextDynamic. It is possible to save certain pages as BMP or PNG files.

There is also the **option to export to multipage TIFF files** - with TextDynamic this requires the "Premium" or "Server" License, with wRTF2PDF TextDynamic Server it requires "Server" License!

Usually this method is used to create monochrome TIFF files for storage and fax.
Those files can also be dithered, to make colored text and images better visible.

Note: When the operation fails a negative value is returned.

Parameters:

PageNr - an Integer. This is the page which should be exported.

filename - a string. The name of the fiel which should be created. (In TIFF mode it has to be "last" when the last page was added)

Xres, Yres - two integers. The resolution for the created bitmap.

Mode - a bitfield.

The first 4 bits are reserved to store the bit depth. Currently only 1 and 24 are supported

the next bits select the export mode

A) - Format Selection
256 - select PNG format
512 - select TIFF format (requires special license).
Create a new file if the filename is <> "". Otherwise add a new page.
otherwise a BMP filer is written

B) Options for TIFFS
1024 - used with TIFF, close the open TIFF file.
You can also pass "last" as filename with Options = 0 .
2048 - used for monochrome TIFF files only. Switch on dithering. (new in V1.61)

C) Rendering options
4096 - also display selection
8192 - do not print watermarks
16384 - do not print header and footer
32768 - do not print images
65536 - display grid lines for tables

Example - Create multipage TIFF file:

```
IWPMemo Memo = wpdllInt1.Memo;

// Open Tiff (Mode=512). Bit depth=1
Memo.GetPageAsBitmap(0, "c:\\rtftotiff.tif", 300, 300, 512+1);

// add all pages (convert until result is negative), select dithering (Mode 2048)
int i = 0;
```

```
while (Memo.GetPageAsBitmap(i, "", 300, 300, 512+2048+1)>0) i++;

// Close the file
Memo.GetPageAsBitmap(0, "last", 300, 300, 1024);
```

4.1.3.2 BB) SetBProp

This method reads and writes internal property flags.

Declaration

```
int SetBProp([In] BPropSel Group, [In] int ID, [In] int Value);
```

Description

This method reads and writes a multitude of internal property flags.

The First parameter selects the group of flags, the second "id" the flag and the third the new value.

If the third parameter "**Value**" is -1 the flag will be cleared,
if it is 1 it will be set,
if it is 0 only the current state (1 or 0) will be returned.

These groups are used:

Group 0 : wpVarBOptions
Group 1 : wpEditOptions
Group 2 : wpEditOptionsEx
Group 3 : wpProtectProp
Group 4 : wpClickableCodes
Group 5 : wpWriteObjectMode
Group 6 : wpViewOptions
Group 7 : wpAsWebPage
Group 8 : wpFormatOptions
Group 9 : wpFormatOptionsEx
Group 10 : wpAcceptFilesOptions
Group 11 : wpPrintOptions
Group 12 : wpClipboardOptions
Group 13 : Security Options.
Group 14 : Global Security Options
Group 15: wpFormatOptionsEx2
Group 16: ViewOptionsEx

Note: Within .NET projects you can use the enum BPropSel to select the group, for example Memo.SetBProp(BPropSel.wpVarBOptions,1,1) to enable the "word wrap" mode.

If you use the method wptextCreateDialog to create the editor window, You can use the bitfields Memo1_SetBProp_ADD.. to set and clear all mentioned flags. Example: param.Memo1_SetBProp_ADD[0] = (1024 * 2) will activate image drag&drop.

New: If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared. (all groups>0)

? Group 0 : wpVarBOptions

Group 0 : Various options

usage: Memo.SetBProp(0, X, -1) to deactivate, Memo.SetBProp(0, X, 1) to activate.

0: Inserting mode on / off = insert vs. overwrite mode
1: WordWrap mode on / off = word wrap to the bounds of the editor
2: SinglePageMode mode on / off = for preview, show one page only
3: CaretDisabled on / off (default: off) = don't display insertion marker
4: Draw table Grid - not a binary property - possible Values are: 0=off, 1=Hide on screen, 2=Hide on printer, 3=Always hide Borders. Use -1 to read current value
5: ShowPagenumber on / off - show page numbers under pages (i.e. thumbnail mode)
Hint: to disable the yellow page number use wpDontDisplayScrollPageHint in ViewOptions: SetBProp(6,27,1)
6: Display text as web page on/off. Also see Group 7 : Webpage Mode.
7: OneClickHyperlink on / off (if on, a single click will trigger the hyperlink event, otherwise the user has to click twice)
8: 3D frame on / off
9: Enabled on / off - accept mouse events
10: Readonly on / off - also see Memo.Readonly for a more complete protection of the text
11: **AcceptFiles** on / off (if on, images can be inserted by drag&drop!). Also see AcceptFilesOptions!
12: Modified yes / no - was text changed ?
13: WantReturns yes / no - accept carriage returns
14: WantTabs yes / no - accept tab key
15: Internally lock the styles against Clear() yes/no
16: The ruler use inch instead of cm yes/no
17: The editor fields on the dialogs use inch instead of cm yes/no
18: Activate the DuplicateWithText mode. The text in a row is not duplicated unless the mode DuplicateWithText is true. (See TextCursor.InsertRow, and AppendRow)
19: Activates the regular popup menu when clicking on a word. This is on by default. If you use the event OnMouseDownWord to show an individual popup dialog please set this property to false. The default value is true.
20: Activate the use of the installed word converter DLLs. They can be selected from the FileOpen dialog. Default = off. If the conversion works depends on the DLL and is not under our control. The application "WordPad" uses this DLLs.
21: Disable the popup / Context Menu.
22: Enable/Disable blinking caret
23: Enable/Disable printing colored or gray borders as dotted black lines to work around printer bug

? Group 1 : wpEditOptions

Group 1 : Options for editing. Also see EditOptionsEx

usage: Memo.SetBProp(1, X, -1) to deactivate, Memo.SetBProp(1, X, 1) to activate.

0: wpTableResizing
1: wpTableOneCellResizing always only one cell like ssCtrl in Shift
2: wpTableColumnResizing // change column width
3: wpTableRowResizing // Change height of row
4: wpClearAttrOnStyleChange //ON: clear the redundant properties when the style name is changed.
5: wpNoAutoWordSelection // don't select complete words (like Word)

6: wpObjectMoving // move images (ObjType=wpobjImage)
7: wpObjectResizingWidth // the width of objects can be changed
8: wpObjectResizingHeight // the height of objects can be changed
9: wpObjectResizingKeepRatio // the width/height of objects can be changed
10: wpObjectSelecting // objects can be selected
11: wpObjectDeletion // objects can be deleted (only used for TWPObject)
12: wpNoAutoScroll // Switch off the new Auto Scroll Feature
13: wpSpreadsheetCursorMovement Cursor up/down in Rows
14: wpAutoInsertRow 0 : wpAutoInsertRow TAB in last cell. Must be combined with0 : wpSpreadsheetCursorMovement
15: wpNoEditOutsideTable to simulate spreadsheet
16: wpBreakTableOnReturnInRow V5: instead of inserting a row break up the table
17: wpActivateUndo activate UNDO
18: wpActivateUndoHotkey
activate ALT + Backspace - requires0 : wpAllowUndo set too
19: wpActivateRedo makes backup of complete text !
20: wpActivateRedoHotkey makes backup of complete text !
21: wpAlwaysInsert don't switch to overwrite
22: wpMoveCPOnPageUpDown Move Cursor on Page up or Down code - V5 ok
23: wpAutoDetectHyperlinks // Create a hyperlink after 'www.' was typed
24: wpNoHorzScrolling
25: wpNoVertScrolling
26: wpDontSelectCompleteField
27: wpStreamUndoOperation // saves also additional info like bands objects ..
28: wpTabToEditFields // used with ProtecteProp: ppAllExceptForEditFields
29: wpSelectPageOnDblClick
30: wpAllowCreateTableInTable // -the create table button allows nested tables

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

? Group 2 : wpEditOptionsEx

Group 2 : Options for Editing - also see "EditOptions"

Also see Group 17: EditOptionsEx2

usage: `Memo.SetBProp(2, X, -1)` to deactivate, `Memo.SetBProp(2, X, 1)` to activate.

0: wpDisableSelection // The user cannot select text (see ViewOptions to hide selection)
1: wpDisableCaret // The caret (insertion point marker at cursor position) is not displayed
2: wpDisableGetFocus // The editor will never receive the focus
3: wpDisableEditOfNonBodyDataBlocks // in Pagelayout mode other DataBlocks (header footer) cannot be selected for edtiting with a click of the mouse
4: wpAllowCursorInRow // the cursor can be placed in row end marker to create a new row with return
5: wpTextObjectMoving // move text objects (ObjType=wpobjTextObj)
6: wpTextObjectSelecting // By defualt allow selectiong of text objects
7: wpNoAutoWordStartEndFieldSelection // Normally a complete field is selected when the cursor is moved over the start or end of a mail merge field. unless wpNoAutoWordSelection is used.
8: wpDisableAutoCharsetSelection // If true the charset is not retrieved by checking the current keyboard layout when the user types
9: wpIgnoreSingleCellSelection // No cell selection by pointing in bottom left corner
10: wpTABMovesToNextEditField // used with forms
11: wpDblClickCreateHeaderFooter // Double click in Margin creates header/footer for all pages.

Use 'OnClickCreateHeaderFooter' event to modify 'range'
12: wpRepaintOnFieldMove // When the cursor moves to a different field the form is repainted. This is important if you use code to highlight the current field. (using event: OnGetAttributeColor)
13: wpKeepCellsWhenCombiningCells // use the HTML way to combine cells horizontally
14: wpAllowSplitOfCombinedCellsOnly // use the HTML way to split cells dont allow split if not combined
15: wpDontClearStylesInNew // If defined Action 'New' will not clear the defined styles
16: wpDontResetPagesizeInNew // If defined clearing the text will not set up the default page size
17: wpSetDefaultAttrInNew // If defined "New" will preset the writing attributes to the default
18: wpAllowDrawDropBetweenTextBlocks
19: wpDisableFastInitOnTyping // Disable the improved typing performance in large paragraphs
20: wpDontTriggerPopupInContextEvent // Changes the time the event OnMouseDownWord is triggered (old behaviour)
21: wpAlwaysColWidthPC // Column Width always in %
22: wpDisableXPosLineUpDown - When using cursor up/down stay in same char pos, not x pos
23: wpDontInitSelTextAttrWithDefaultFont - Display empty font / size selector when current attributes are default attributes
24: wpDontSelectCellOnSpreadsheetMovement - Do not select next cell when using TAB in a table
25: wpScrollLineWise when suing scroll bar up/down button scroll line wise
26: wpZoomWithMouseWheel - zoom in and out when pressing CTRL while using mouse wheel
27: wpTableRowResizingWithCTRL - allow table row resizing only when also CTRL is pressed (overrides wpTableRowResizing in group 1).

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

? Group 3 : wpProtectProp

Group 3 : Select which text or which objects should be protected

usage: Memo.SetBProp(3, X, -1) to deactivate, Memo.SetBProp(3, X, 1) to activate.

```
0: ppParProtected // the complete paragraph cannot be deleted --> WPAT_ParProtected
1: ppCheckAllText // Trigger event For the complete Text
2: ppAllExceptForEditFields // Only allow editing In edit fields. Cursor jumps between ed
3: ppNoNewParagraphsInEditFields // Do Not allow par insertion/deletion In EditFields
4: ppProtected // Text With the style afsProtected cannot be overridden
5: ppHidden // Text which uses the style 'hidden'
6: ppIsInsertpoint // Protect the field start Or End markers (wpobjMergeField)
7: ppIsMergedText // used To be 'ppAutomatic' The oposite of : ppAllExceptForEditFields
8: ppIsBookmark // Protect the bookmark start Or End markers (wpobjBookmark)
9: ppIsTextObject // Protect the fields (such As PAGE numbers) (wpobjTextObj)
10: ppIsImageObject // Protected the images (wpobjImage)
11: ppProtectionTextObjects // used With TWPTextObj.Mode 'wpobjWithinProtected'
12: ppIsInvisible // invisible Text is protected
13: ppAllowEditAtTextEnd // Allow typing at Text End
14: ppProtectSelectedTextToo // normally selected Text is Not protected, wit this flag it
15: ppDontProtectAttributes // V5.15 - If defined it is possible To change the attributes
16: ppDontUseAttrOfProtected // Do Not use the charattr If the Text is protected
17: ppNoEditAfterProtection // Do Not allow editing at the End of the paragraph If the la
18: ppInsertBetweenProtectedPar // Allow the insertion between protected paragraphs
19: ppIsTextObjectCustom - similar To ppIsTextObject which it overrides - but
            Do Not protected page And Date objects. This is useful To protect custom obje
20: ppBookmarkKeepStructure // Used With ppIsBookmark. When Text is deleted which contains
```

```
            the bookmark objects are recreated To Preserve the logical structure of the d
21: ppInsertpointKeepStructure // Used With ppIsInsertpoint - reserve the merge fields.
22: ppIsHyperlink //  Protect the hyperlink start Or End markers (wpobjHyperlink)
23: ppNoEditBeforeProtection // Disable insertion If protected Text is first In paragraph
```

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

? Group 4 : wpClickableCodes

Group 4 : Select which object types are used as "hyper links".

usage: Memo.SetBProp(4, X, -1) to deactivate, Memo.SetBProp(4, X, 1) to activate.

1: wpobjMergeField // merge fields
2: wpobjHyperlink // [default] - hyperlink objects
3: wpobjBookmark // bookmarks

Note, the objects which can be used to mark clickable text (= hyperlinks) area always used in pairs, such as <a>...

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

? Group 5 : wpWriteObjectMode

Group 5 : Change the way embedded images are saved

usage: Memo.SetBProp(5, X, -1) to deactivate, Memo.SetBProp(5, X, 1) to activate.

0: wobDontSaveImages // Images are not saved
1: wobRTF - best mode to save images (default)
2: wobRTFNoBinary - do not save as binary (use hex code)
3: wobStandard - use proprietary format
4: wobStandardNoBinary - use proprietary format in ASCII mode
5: wobStandardAndRTF - write images in both modes (double) - not recommended

If the second parameter id=-1 and the value =-1 the value "wobRTF" will be selected.

? Group 6 : wpViewOptions

Group 6 : Change the way the text is displayed in the editor. Also see ViewOptionsEx.

If you use a split screen editor both editors can use different modes.

usage: Memo.SetBProp(6, X, -1) to deactivate, Memo.SetBProp(6, X, 1) to activate.

0: wpShowGridlines - show the grid lines for tables
1: wpDisableHotStyles - do not use any fly over effects for links
2: wpShowCR - display "¶" symbol - also see wpShowCRButNotInCells.
3: wpShowFF - display symbol for hard page breaks
4: wpShowNL - display symbol for line breaks Char(10)
5: wpShowSPC - display dots for spaces
6: wpShowHardSPC - display dots for hard spaces Char(160)
7: wpShowTAB - display arrows for tabstops
8: wpShowParCalcNames // Display the names assigned using property WPAT_PAR_NAME

9: wpShowParCalcCommands // Display the formulas
10: wpShowParNames // Display the names assigned using property TParagraph.Name
11: wpNoEndOfDocumentLine // If enabled a line will be displayed in normal mode at the bottom. (Ignore the "no")
12: wpHideSelection // Selection possible but not displayed
13: wpHideSelectionNonFocussed // selection not displayed when not focussed
14: wpHideSelectionNonFocussedAndInactive // Selection not displayed if not active (not connected to toolbar)
15: wpTraditionalMisspellMarkers [default] Draw "Word like" curly underlines (default)
16: wpDisableMisspellMarkers // Do not draw curly underlines
17: wpShowPageNRinGap // Display a page number between the pages (only when using the page gap mode)
18: wpDrawFineUnderlines // Underlines are always one pixel
19: wpDontGrayHeaderFooterInLayout // header and footer texts are not grayed out. (more: header&footer)
20: wpInfiniteTextArea // When scrolling over the end of the document the start is displayed agaian
21: wpDontPaintPageFrame // Do not draw the rectangle around page
22: wpCenterPaintPages // center the pages in the window horizontally
23: wpUseOwnDoubleBuffer // Usually a shared double buffer is used, this deactivates the caching
24: wpDrawPageMarginLines // mark the page margins on the virtual page
25: wpDontDrawSectionMarker // don't draw an arrow on the left side when a section starts
26: wpDrawHeaderFooterLines // draw a rectangle round header and footer areas
27: wpDontDisplayScrollPageHint // if set, don't display page number hint when using scrollbar
28: wpDontDrawObjectAnchors // do not draw the anchors for a movable object
29: wpShowCRButNotInCells [default] - similar to wpShowCR but don't display "¶" in tables cells

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

? Group 7 : Webpage Mode

Group 7 : Format the text using a special routine which is optimized for HTML.

It will not preserve all features which are possible in RTF, such as aligned or movable images.

usage: `Memo.SetBProp(7, X, -1) to deactivate, Memo.SetBProp(7, X, 1) to activate.`

Flags:

0 : activate/deactivate the "as webpage" display.

When this mode is activated a different formatting procedure is used to measure the tables more like a webbrowser. Not all text features are supported in the webpage mode.
RTF2PDF Tip: You can also use **RtF2PDF1**.Command(1312, 1).

Note: You can use TextCommandStr(6,"..") to enable the provided HTTP DLL to load linked images.

Refine the "as webpage" mode using this flags:

1 : wpNoMinimumWidth - on/off

2 : wpNoMinimumHeight - on/off

reserved for future: 3 : wpLimitToPageWidth - on/off - **default on**

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

C# Example:

```
// Activate HTML Formatting
memo.SetBProp(WPDynamic.BPropSel.wpViewOptionsEx, 0, 1);
```

? Group 8 : wpFormatOptions

Group 8 : Modify the formatting routine. Also see FormatOptionsEx.

usage: Memo.SetBProp(8, X, -1) to deactivate, Memo.SetBProp(8, X, 1) to activate.

0: wpDisableAutosizeTables // Default - we suggest to enable this flag. Otherwise tables imported from RTF can look wrong
1: wpNoMinimumCellPadding // Do not add a one pixel padding to all cells
2: wpfDontBreakTables // *1) do not break tables at all. Also see : wpKeepTogetherAdjacentTables in FormatOptionsEx
3: wpfDontBreakTableRows // do not break table rows
4: wpDontClipCells // do not clip cells if absolute row heights are used
5: wpfIgnoreMinimumRowheight // Do not use the row height property
6: wpfIgnoreMaximumRowheight // Do not use the row maximum height property
7: wpTableRowIndicator // *2) must be combined with EditOptionsEx : AllowCursorInRow
8: wpfIgnoreKeep // *3) The WPAT_ParKeep property is ignored (do not break par)
9: wpfIgnoreKeepN // The WPAT_ParKeepN property is ignored (do not break adjacent par)
10: wpfKeepOutlineWithNext // Text which uses the WPAT_ParIsOutline property is kept with the next text
11: wpfAvoidWidows // Avoid single lines on old page
12: wpfAvoidOrphans // Avoid single lines on new page
13: wpfCenterOnPageVert // Center text on all pages
14: wpfHangingIndentWithTab // V5: first tab in paragraph jumps to indent first (this always happens if no tabs are set!) // the left indent will be handled as first tabstop not only if the tab is the // first character but also if the text before the tab fits into the first indent. (="Word" like)
15: wpfDontTabToIndentFirst // The oposite to : aFormatOption([wpfHangingIndentWithTab
16: wpJustifySoftLinebreaks // Justify \n
17: wpJustifyHardLinebreaks // Justify \r
18: wpUseHyphenation // Use soft hyphens in the text (inserted with Ctrl + '-')
19: wpFooterMinimumDistanceToText // The footer texts start after the body text and does not keep a certain distance to the text
20: wpShowBookmarkCodes // display bookmark tags
21: wpShowHyperlinkCodes // display hyperlinks tags
22: wpShowSPANCodes // display SPAN tags
23: wpShowInvisibleText // show text which would be otherwise hidden
Experimental flags:
24: wpfHideEmptyParElements // reserved: Can be used when editing HTML files with nested DIV elements
26: wpWriteRightToLeft // activate RTL (right-to-left) writing.
27: wpAutoWriteRightToLeft // reserved
32: wpActivateRTLMode // if active, the whole text uses RTL writing

Troubleshooting flags
25: wpfXMLOutlineMode // Debugging mode: Show paragraph tree similar XML in IE
28: wpUseAbsoluteFontHeight // Calculate the height of the text using the font size alone

29: wpfAlwaysFormatWithScreenRes // .. even if RM600 is defined in WPCtrMemo
30: wpDisableSpeedReformat // format only the current page and the next page on regular input
31: wpDontAdjustFloatingImagePosition // do not adjust image position to keep it on the page

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

Note:
*1) Breaking up table rows and tables on different pages can be generally deactivated. If a table does not fit on next page, it will be divided anyway.
*2) When the cursor is at the end of a row, a new row can be created by pressing "Enter"
*3) "Keep" control avoids breaking up certain paragraphs on several pages.
It tries to keep paragraphs together. KeepN keeps this paragraph with the next on one pages.

? Group 9 : wpFormatOptionsEx

Group 9 : Modify the formatting routine.

usage: `Memo.SetBProp(9, X, -1)` to deactivate, `Memo.SetBProp(9, X, 1)` to activate.

0: wpDontAddExternalFontLeading // When measuring the font height don't add the // value defined for a font: Metrics.tmExternalLeading. This improves compatibility to MS Word
1: wpfKeepTablesInTextArea // If defined avoid that table go into right margin
2: wpKeepTogetherAlwaysNewPage // if : wpfDontBreakTables is used a table which is too large will also create a new page
3: wpKeepTogetherAdjacentTables // When tables are not separated by paragraphs they are kept together as well
4: wpDontUseTablePadding // Do not read the padding of cells from table
5: wpfIgnoreVertAlignment // Switches off the vertical alignment
6: wpfKeepNUsesParImages // When using KeepN paragraph aligned images will be kept on same page as their anchor paragraph.
7: wpDontIgnoreSpacebeforeOnTopOfPage // switch "Word" mode off
8: wpfDontHideParAboveNestedTable
9: wpDontUseRowPadding // Do not read the padding of cells from table row
10: wpDontUseBorderPadding // Do not use the border width to calculate padding
11: wpDontInheritCellAttr // Dont inherit cell attributes from table row and table
12: wpDontUseSPANStyles // Do not use the stalyes of hyperlink and SPAN objects . Speeds up reformat a lot!
13: wpfDisableJustifiedText
14: wpAlwaysContinueBorderAfterCR // Paragraph.SplitAt (= ENTER key in editor) will also copy border attributes to // the new paragraph if the split position is at the end
15: wpNoDefaultParStyles // Disable default paragraph style handling (So you can handle // it yoursel in the OnInitializePar event)!
16: wpfAutoRestartSimpleNumbering // Reset a numbering level after a line which was not numbered.
17: wpfSimpleNumberingPriorityOverOutlines // this only affects numbering which does not use outlines
18: wpfNoAutoDecTabInTable // Do not automatically align to only decimal tabstop in table cells
19: wpfNoTableHeaderRows // Ignore table header
20: wpfNoTableFooterRows // Ignore table footer // Experimental Flags
21: wpAllow_WPAT_NoWrap // If active don't wrap par marked with WPAT_NoWrap
22: wpfMixRTLText // If a paragraph which uses the paprRightToLeft attribute the // western text is not reordered
23: wpfStoreWPObjectsInRTFDataProps // unless this flag is used the TWPObjects will // be stored in the TWPRTFDataCollection and not in the TWPRTFDataProps // Activate User CharStyle

24: wpCharEffectIsUserAttribute // Currently the character attribute WPAT_CharEffect is not used. // It can be used to store custom information. If the flag wpCharEffectIsUserAttribute is used // the RTF engine will ignore this property.
25: wpfIgnoreTrailingEmptyParAtFooter // Empty paragraphs are ignored at´the footer end
25: wpfDontCombineDifferentStylePars // Empty paragraphs are ignored at´the footer end
26: wpfNestedNumberingInTableCells // Empty paragraphs are ignored at´the footer end

27: wpfNestedNumberingInTableCells
28: wpfNumberingControlledByStyles
29: wpfSectionsRestartFootnoteNumbers
30: wpfDontIgnoreKeepNInTable

Note: RTF2PDF uses this default flags
wpfIgnoreKeepN, wpfAvoidWidows, wpfAvoidOrphans, wpUseHyphenation, wpNoMinimumCellPadding, wpDisableAutosizeTables

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

? Group 10 : wpAcceptFilesOptions

Group 10 : Change the way files are handled when dropped on an editor.

This mode is only used when "AcceptFiles" has been switched on using SetBProp(0, 11, 1)

usage: `Memo.SetBProp(10, X, -1)` to deactivate, `Memo.SetBProp(10, X, 1)` to activate.

0: wpDropCreatesLinkedImage // Only the pathname of the image will be stored
1: wpDropCreatesMovableParObject // The image will be positioned relatively to paragraph
2: wpDropCreatesMovablePageObject // -or- The image will be positioned relatively to page
3: wpDropCreatesNoWrapImage

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

? Group 11 : wpPrintOptions

Group 11 : Modify the way printing is performed

usage: `Memo.SetBProp(11, X, -1)` to deactivate, `Memo.SetBProp(11, X, 1)` to activate.

0: wpDoNotChangePrinterDefaults
1: wpIgnoreBorders
2: wpIgnoreShading
3: wpIgnoreText
4: wpIgnoreGraphics
5: wpUsePrintPageNumber
6: wpDontPrintWatermark
7: wpDontReadPapernamesFromPrinter
8: wpAlwaysHideFieldmarkers //hide field markers in PrintDialog
9: wpDontAllowSelectionPrinting
10: value=-1: read, otherwise the duplex mode. Possible values: 0=Don't change duplex,

1=None, 2=Horizontal, 3=Vertical
11: wpCompareWidthHeightToDetectLandscape (default=on)
12: wpAllColorsAreBlack - print colors in gray scale

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

? Group 12 : wpClipboardOptions

Group 12 : Modify the clipboard is used. Here also Drag&Drop is customized

usage: `Memo.SetBProp(12, X, -1)` to deactivate, `Memo.SetBProp(12, X, 1)` to activate.

0: wpcoNoInternalDragAndDrop //Switch off the internal drag&drop
1: wpcoNoDragAndDropFromOutside // Allow only Drop from within
2: wpcoAlwaysDeleteInDragSource // Delete dragged source text if from different editor too!
3: wpcoDontMoveCursorDuringDrag // Do not show cursor when dragging
4: wpcoNoAutoSelSpaceExtension // Don't select preceding or trailing spaces when doing Drag&Drop
5: wpDontHideCaret // do not hide the caret while selection is active

Enable/Disable certain formats:

6: wpcoDontPasteWPT // Don_t paste the internal proprietary WPT format
7: wpcoDontPasteRTF // don't paste ANSI
8: wpcoDontPasteANSI
9: wpcoDontPasteUNICODE
10: wpcoPasteHTMLWhenAvailable // not recommended - paste HTML when availbale
11: wpcoDontPasteGraphics // don't paste graphics (unless embedded in RTF text)
12: wpcoDontCopyRTF // Do not copy RTF
13: wpcoDontCopyANSI
14: wpcoDontCopyUNICODE
15: wpcoDontCopyWPT

This flags change the way pasted text is handled:

16: wpcoPreserveBorders // preserve the borders at the destination
17: wpcoPreserveShading // preserve the shading at the destination
18: wpcoPreserveIndents // preserve the indents at the destination
19: wpcoDontAutoAppendSpace // do not append a space
20: wpcoPasteAsNestedTable // cells are always pasted as nested in table
21: wpcoDontPasteFonts // keep current font name
22: wpcoDontPasteFontSizes // keep current font size
23: wpcoDoNotUseInsertMode // Any paste operation will completely replace the document!
24: wpcoDontCopyProtectedText
25: wpcoDontCopyProtectedText

Some additional flags

26: wpcoAlwaysCopyImagesEmbedded // Also linked images are embedded in clipboard
27: wpcoAlsoCopyHTML // Copy HTML
28: wpcoAlsoPasteRTFVariables // Modify the document variables when pasting text
29: wpcoDontPasteWhenTextIsSelected // Don't allow pasting when text is selected
30: wpcoPastedANSIDoesNotInheritParAttr // pasted ANSI text does not use current indent and other paragraph attributes

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

? Group 13 : Security Options.

Group 13 : Security Options.

This flags allows it to avoid that the user copy or saves text from this editor.

usage: Memo.SetBProp(13, X, -1) to deactivate, Memo.SetBProp(13, X, 1) to activate.

1 : DontUseGlobalSettings - this editor should not use the global settings (group 14).
2 : Do not allow any paste operation
3 : Do not allwo any copy operation
4 : Do only allow copy and paste within the application, paste in other application is not possible. (The clipboard contents is encrypted with a dynamic password)
5 : Save dialogs and buttons are disabled.
6 : Load dialogs and buttons are disabled.
7 : Print dialogs and buttons are disabled.
100: Set flag 4, 5 and 6 and disable any subsequent change to this property.

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

? Group 14 : Global Security Options

Group 14 : Global Security Options.

usage: Memo.SetBProp(13, X, -1) to deactivate, Memo.SetBProp(13, X, 1) to activate.

This flags allow it to avoid that the user copy or saves text from all TextDynamic editors.

1 : not used
2 : Do not allow any paste operation
3 : Do not allow any copy operation
4 : Do only allow copy and paste within the application, paste in other application is not possible. (The clipboard contents is encrypted with a dynamic password)
5 : Save dialogs and buttons are disabled.
6 : Load dialogs and buttons are disabled.
7 : Print dialogs and buttons are disabled.
100: Set flag 4, 5 and 6 and disable any subsequent change to this property.

You can also use Command(WPDLL_COM_PROTECTEDTEXT=9119) to apply the global flags:

```
WPDLLInt1.Command(9119,100,0)
```

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

.

? Group 15: wpFormatOptionsEx2

Group 15 : Modify the formatting routine.

usage: Memo.SetBProp(9, X, -1) to deactivate, Memo.SetBProp(9, X, 1) to activate.

```
 1: wpfSectionsRestartPageNumbers // restart page numbering In sections
```

```
 2: wpfUseKerning // optimize the Text formatting And display (deafault=off)
 3: wpfCodeObjectAsXMLTags // used With the special "tag" XML reader.
 4: wpDisableBalancedColumns // don't use the new column balancing
 5: wpfCodeObjectCheckParStyles // code objects format the contained text according to sty
 6: wpfNewKeepNSupport //(Default =on) better Keep N Support which also handles tables. D
 7: wpfHideParagraphWithHiddenText //(Default =off) If a paragraph contains nothing but h
    // this is also true, if it is empty and its default attribute contains the text attr

 8:  wpAllowFullWidthImagesInHeaderFooter // allow images in header and footer which are
 9:  even if they are not full page size
10: wpMarginMirrorBookPrint // makes V6 work like V5
11: wpIgnboreAllTabsInPageMargin // WPTools 6 will ignore all tabs after the right margi
12: wpAlwaysContinueShadingAfterCR // If true, an enter at first position will not remov
```

If the second parameter id=-1 and the value =-1 the complete set of flags will be cleared.

? Group 16: ViewOptionsEx

Group 16 : Change the way the text is formatted and displayed in the editor. Also See ViewOptions.

If you use a split screen editor both editors can use different modes.

usage: Memo.SetBProp(16, X, -1) to deactivate, Memo.SetBProp(16, X, 1) to activate.

```
 0: wpAlwaysDrawImageAnchors - always draw the thumb tacks for movable images
 1: wpAlwaysDrawTextboxAnchors - always draw the thumb tacks for text boxes
 2: wpAutoThumbnailMode - when thumbnail mode is active, use special logic for clicking
 3: wpDisableAllSpecialAttributes - do not highlight hyperlinks, fields etc (See Special
 4: wpDisableCharacterStyles - ignore character styles (RTF \cs tag)
 5: wpDisableColorInTableOfContents - a table of contents will be printed in black, no m
 6: wpDontExtendSelectionToRightMargin - when selection text, the highlight color will n
 7: wpInvertActiveRow - Invert the cells of the current row
 8: wpInvertActiveRowExceptForProtected - Do not invert protected cells

 9:  wpShowCurrentCellAsSelected - Displays current cell to be selected. Disables curren
10: wpShowCurrentRowAsSelected - Displays current table row to be selected. Disables cu
11: wpShowCurrentTableAsSelected - Displays current table to be selected. Disables curr
12: wpCursorPreview, // Draw a dotted line where the cursor will be placed when the use
13: wpUnderlineWebLinks // Underline http, https and www

14: wpHideParBordersBeforAndAfterPageBreaks, // Hide borders at page breaks
15: wpShowColumns, // Only for Premium - draw rectangles a round columns (are not print
16: wpShowColumnCenterLine, // Only for Premium - draw a line in the middle between all
17: wpDontHideUnderlineAtLineEnd, // Don't hide underline under spaces in justified tex
18: wpDisable2PassPaint // Disable the 2 pass paint system which first paints character
```

? Group 17: EditOptionsEx2

Group 2 : Options for Editing - also see "EditOptions" and "EditOptionsEx"

usage: Memo.SetBProp(17, X, -1) to deactivate, Memo.SetBProp(17, X, 1) to activate.

```
 0: wpDontAllowDragTextInFields - do not allow to drag text into merge fields
```

```
1: wpDontAllowDragImagesInFields - do not allow to drag images into merge fields
2: wpEnableDragFieldsInFields - do not allow to drag fields into merge fields. Default =
3: wpDisableDragDropInShowFieldsMode - if ShowFields is active, disable drag&drop into f
4: wpDisableDoubleclickSelectsMarker - do not select field marker on double click
5: wpSuppressReplaceWordsMessage - when doing search&replace, do not show the message "x
6: wpAlwaysTriggerFieldFocusEvent - the events OnFieldEnter/OnFieldLeave will be trigger

7: wpRowMultiSelect, // In Table row multi select with CTRL
8: wpDeleteAtEOFRemovesSpaceAfter, // DELETE in last line of text sets spaceafter to 0
9: wpCellMultiSelect // In Table cell multi select with CTRL
10: wpDisableSelectAll // SelectAll cannot be executed anymore. Thats useful when workin
11: wpClearCellsOnDelete // when cells are selected in a table they will always be clear
```

4.1.3.2 BC) SetIProp

This function sets properties of the editor.

Declaration

```
int SetIProp([In] int ID, [In] int Value);
```

Description

Using an ID and a property value certaion properties can be read and set. If the property value is -1 the current value will be read but no new value will be set.
This IDs are defined:
0: not used
1: Set TextCursor. (Param = enum MemoCursor)
2: Set hyperlink cursor. (Param = enum MemoCursor)
3: Set cursor for text objects (Param = enum MemoCursor)

Example C#:

```
// This code sets the cursor used for hyperlinks:
  wpdllInt1.Memo.SetIProp(
  (int)WPDynamic.MemoIProp.HyperlinkCursor,
  (int)WPDynamic.MemoCursor.HandPoint);
```

4: Change spell check strategie:
0= CheckInInit, - Words are checked when the paragraph is initialized. The complete text is checked after loading
1= CheckInPaint - (default) Words are checked before a paragraph is painted.
2= CheckInInitAndPaint - check whole text first

5: Set X offset from text to border of editor to the left

6: Set Y offset from top border to first page

7: Check if the toolbar is currently visible. Use Value = -1. Result =1 if visible, otherwise 0.

8: Set the minimum width of the caret. Default = 1

9: Set the default text color (default = black)

10: Set the background color (ColorPaper and ColorDesktop)

11: Select search direction (TextCursor.FindText):
a value >=0 selects forward search, -1 select backwards search.

Example:

```
wpdllInt1.Memo.SetIProp(9, 255);
```

```
wpdllInt1.Memo.SetIProp(10, 0);
```

Example VB6:

```
WPDLLInt1.Memo.LayoutMode = wplayNormal
WPDLLInt1.Memo.SetIProp 5, 70   ' x-offset from left to text
WPDLLInt1.Memo.SetIProp 6, 70    'y-offset from top to text
```

4.1.3.2 BD) SetWYSIWYGMode

Declaration

```
procedure SetWYSIWYGMode(Mode: Integer)
```

Description

This mode changes the rendering and formatting:

Mode=0 **- best option for perfect printing**

Default - format to printer, do not use special "Kerning" Mode

Mode=1:

format to printer and use special "Kerning" Mode

Mode=2 **- best option for perfect screen display**

format for screen, do not use special "Kerning" Mode

Mode=3:

format for screen and use special "Kerning" Mode

Mode = 4 - best option for good printing and good display

Change the formatting resolution to work best with screen and printer. This also changes the line breaks, so we decided to not make it the default mode.

Mode =10 / -10:

Activate / Deactivate the improved character spacing added to V1.54 (default = on). Only affects Mode 0.

4.1.3.2 BE) UpdateDialog

Declaration

```
function UpdateDialog(DialogID: Integer; ParmID: Integer; IntValue: Integer;
      const StrValue: WideString; Mode: Integer): Integer
```

Description

The method is reserved to modify the internal dialogs.

Please note, the dialogs can be localized using the XML script in side the "PCC" file. Please see SetLayout and WPImagePack Application.

Currently supported is:

DialogId = 1, modify the preview dialog

ParamID = 0 - select the toolbuttons

IntValue=1: Show the buttons

IntValue=2: Hide the buttons except for Close and Print

IntValue=0: Hide the buttons, except for "X" to close

4.1.3.2 BF) SetRTFVariable

Declaration

```
void SetRTFVariable([In] string Name, [In] string Value);
```

Description

Sets the value of a certain string variable which will be stored with the document.

Category

Document Properties

4.1.3.2 BG) Statistic

Declaration

```
void Statistic([In] int Mode, out int Pages, out int Paragraphs, out int Lines,
out int Tables, out int Rows, out int Images, out int Words, out int Letters,
out int Characters);
```

Description

The method Statistic() can be used to retrieve information about the loaded text.

VB6 Example:

```
  Dim Pages As Long  ' Count of pages
  Dim Paragraphs As Long ' Count of Paragraphs, also table cells
  Dim Lines As Long ' Count of lines
  Dim Tables As Long ' Count of tables
  Dim Rows As Long ' Count of rows in all tables
  Dim Images As Long ' Count of Images (but not text boxes)
  Dim Words As Long ' Count of words.
  Dim Letters As Long ' Count of characters which are not word delimiters
  Dim Characters As Long ' Count of characters. Optional adjacent spaces are counted as one

  WPDLLInt1.Memo.Statistic 0, Pages, Paragraphs, Lines, Tables, Rows, Images, Words,
Letters, Characters
```

Parameters	
Mode	bit 1: do not count in body texts

	bit 2: do not count in header texts bit 3: do not count in footer texts bit 4: do not count in footnotes bit 5: do not count in text boxes bit 6: adjacent spaces are counted as one "Character"
Pages	Count of pages
Paragraphs	Count of Paragraphs, also table cells
Lines	Count of lines
Tables	Count of tables
Rows	Count of rows in all tables
Images	Count of Images (but not text boxes)
Words	Count of words
Letters	Count of characters which are not word delimiters
Characters	Count of characters. Optional adjacent spaces are counted as one

Category
Display Status Information

4.1.3.2 BH) Tables_WidthFixed

Declaration
```
void Tables_WidthFixed();
```

Description
Process all tables in the text to change any variable column width to a fixed width (twips).

Category
Table Support

4.1.3.2 BI) Tables_WidthPC

Declaration
```
void Tables_WidthPC([In] bool Enable);
```

Description
Process all tables in the text to change any fixed column width to a variable width (%).

Category
Table Support

4.1.3.2 BJ) TextCommand

Declaration
```
int TextCommand([In] int ComID, [In] int paramA, [In] int paramB);
```

Description
This method has been reserved for custom enhancements.

Commands 1-10: TOC, DefaultAttr
Commands 11-23: SyntaxHighlight, Selection, Background Gradient
Commands 21-23: Tabstops
Command 24: Draw lines, Page Size

? Commands 1-10: TOC, DefaultAttr

Command ID: 1
Modify the width of the tables in the document. If bit 1 of parameter A is set, all tables will be modified, otherwise only the tables in the text body. Both width properties of the tables will be deleted. If bit 1 of parameter B has been set, the percentage width parameter (WPAT_BoxWidth_PC) will be set to 100%.

Command ID: 2
This command inserts a table of contents at the current position or the position which was marked with a merge field with the name __TOC__.
Using the parameter A you can control the way the TOC is created: bit 1: also process paragraphs in tables, bit 2: create hyperlinks, bit 3: do not use text within bookmarks but after bookmarks on the same paragraph, bit 4: do not use the TOCOutlineLevel attribute, simply collect text which is marked with bookmarks named "_Toc*", bit 5: do not create page numbers. If bit 1 is set in parameter B the text will not be formatted automatically.

Command ID: 3:
Asks the user to save the document if it was changed. If it was not changed or saved the result value is 1, othewise it is 0.

Command ID: 4:
Set the MaxParLength property to parameter A. If > 0 than the line break is fixed at this count of characters (unless the page width is smaller).

Command ID: 5:
Update visible cursor position. Scrolls if necessary.

Command ID: 6:
Reads and optionally sets the default attribute index for this text.

ParamA is used as bit field: 1=set the attribute index in Param B. 2=lock the default attributes, 4=unlock the default attribute.

```
// Init default font!
wpdllInt1.AttrHelper.Clear();
wpdllInt1.AttrHelper.SetFontface("Courier New");
wpdllInt1.AttrHelper.SetFontSize(12);
wpdllInt1.Memo.TextCommand(6,3, wpdllInt1.AttrHelper.CharAttrIndex );
```

Command ID: 8:
Selects the cell the cursor is currently located within. If paramA=1 the selection will be added to the already made selection.

Command ID: 9 - Internally used

Command ID: 10 - Internally used

? Commands 11-23: SyntaxHighlight, Selection, Background Gradient

Command ID: 11
The result is 1 if currently text is selected.

Command ID: 12
Selects the internal Syntax Highlighting
paramA = 0 - disables the highlighting,

paramA = 1 - selects the HTML / XML highlighter. (Attributes are applied to the text)

Obsolete - more functionality has been implemented with TextCommandStr!
Please see TextCommandStr ID 16 - Syntax Highlighting for more features and other modes.

Command ID: 13
Combines all adjacent tables. Returns number of combined tables.

Command ID: 14
Tries to create merged cells for cells which are wider than the others in a column. Returns the count of changed tables.

Command ID: 15
See Command ID 15 - Update ShowFields Mode

Command ID: 16
Forces the update of the attributes provided by CurSelAttr

Command ID: 17
Returns the number of milliseconds which elapsed since the completion of the last user keyboard or mouse interaction. This can be useful to start background tasks only in idle phases.

Command ID: 18 - reformat
Starts the reformat of the text in the next idle phase

ParamA = 0: Works like Memo.Reformat

ParamA can be used to force the editor to formatting of all paragraphs:
if bit 1 is set: Format all paragraphs
if bit 2 is set: Initialize all paragraphs
if bit 3 is set: The "SpellAsYouGo" markers in all paragraphs are removed

Command ID: 19
Returns the number of items on the undo stack.

Command ID: 20
Sets the second color "ColorDesktopTo" to draw a gradient in the background. The start color is ColorDesktop.
If the parameter paramB is <> 0, the gradient will be horizontal.

? Commands 21-23: Tabstops

Command ID: 21 (also see Tabstop Category)
Adds a tab stop. This method works with selections or the current paragraph.
paramA = position in twips
paramB, low byte, the mode (left, right, center, decimal)
paramB, high byte, the fill mode (no fill, dots, MDots, hyphen, underline, THyphen, EqualSign, Arrow)

Command ID: 22
Deletes the tab stop at position paramA

Command ID: 23
Clears all tabs tops in the selected text or the current paragraph.

? Command 24: Draw lines and background, Page Size, Force Pagecount

Using this command various page size and drawing options can be changed.

paramA selects the id of the option, paramB changes it.

1) Set the current and, at the same time, the default page size.

ParamA can be one of this IDs, paramB is expected to be a twip value.
WIPR_PAGE_WIDTH = 1;
WIPR_PAGE_HEIGHT = 2;
WIPR_PAGE_LEFT_MARGIN = 3;
WIPR_PAGE_RIGHT_MARGIN = 4;
WIPR_PAGE_TOP_MARGIN = 5;
WIPR_PAGE_BOTTOM_MARGIN = 6;
WIPR_PAGE_HEADER_MARGIN = 7;
WIPRPAGE_FOOTER_MARGIN = 8;
Alternatively the page size can be changed using the interface PageSize.

2) Select the current page drawing mode:

a) Select the mode:
WIPR_ED1_PAGE_DRAW_MODE = 9;

This is a bitfield. The following values are possible and be combined:
1: draws one bitmap in the background (Watermark) - this can be toggled using "wpaShowWatermark".

2: draw two bitmaps in the background, but only on screen. This can be used to draw a simulated notebook.
The images which are drawn can be selected using with the Background Image Commands. Use action wpaShowBackgroundImage to change the state.

4: draw a rectangle or lines around the page.
The display can be toggled using the action wpaShowAddressArea.
You can select the line *mode* using WIPR_ED1_PAGE_LINE_MODE.

8: draw and print a folding line at 1/3 of the page. Use Action wpaShowFoldLine.

32: draw and print a line through the unused area at the bottom of a page. Action wpaShowInvalidateAllPages toggles the option.

64: draw and print a line through the unused area at the bottom of last page. Action wpaShowInvalidateLastPage toggles the option.

128: Draw the page number in the lower right corner.

b) Select the line mode ("Address Area")
WIPR_ED1_PAGE_LINE_MODE = 10; // 0=adress top-left, 1=addres top right, 2=lineoffset

ParamB can be
0 : Draw a rectangle at the top left area
1 : Draw a rectangle at the top right area
2 : Draw lines in a certain offset of the borders of the page. (Don't mix up with the margins)
3 : Draw lines in a certain offset of the borders of the page.
Exchange the left and right offset for even pages.

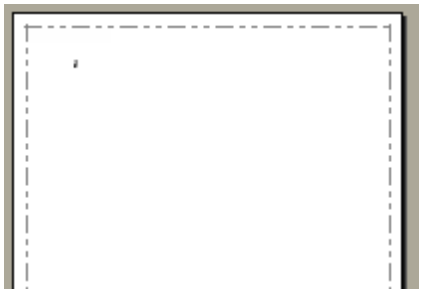
4: Draw the lines only on the first page.

Use this IDs to set the offset values in twips. If a value remains 0, the line is not drawn.
WIPR_ED1_PAGE_LINEOFFSET_LEFT = 11;
WIPR_ED1_PAGE_LINEOFFSET_RIGHT = 12;
WIPR_ED1_PAGE_LINEOFFSET_TOP = 13;
WIPR_ED1_PAGE_LINEOFFSET_BOTTOM = 14;

Note: You need to use Command(24, 9, 4) to activate the display.

Example: Draw lines - all offsets at 360

```
Memo.TextCommand(24,
Memo.TextCommand(24,
Memo.TextCommand(24,
Memo.TextCommand(24,
Memo.TextCommand(24,
Memo.TextCommand(24,
```



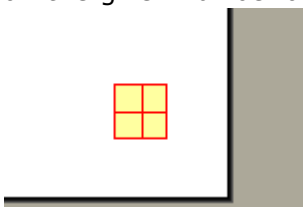
3) Force a certain page count in the editor

a) WIPR_ED1_PAGES_MINCOUNT = 15

Forces the text to have at least the given number of pages. The added pages are drawn empty.

b) WIPR_ED1_PAGES_MAXCOUNT = 16

Any additional pages than the given number are not displayed.
On screen a "plus sign" is displayed if the page count exceeds the count of displayed pages:



c) WIPR_ED1_PAGES_MULTIPLICATOR = 18

Forces the page count to be dividable by the given value. If required, empty pages are appended.

Please note, that a second editor can show the same text with a different count of pages.
For example the thumbnail view will display less then X pages unless You also execute the same code for "Memo2".

4) Read the current page Count

Use either TextCursor.PageCount or this command with the id WIPR_ED1_PAGECOUNT = 17

Note:
In case You cannot use the COM Interface IWPMemo, this feature is also available by sending the windows message 1031 to the editor.
Parameters wparam as paramA, lparam as paramB.
If you need to modify editor#2, just add the value 100 to wparam.

Also see "TextDynamic as Popup Editor (method wptextCreateDialog)".

? Command 25: Select Watermark Image

You can use Command(143,..) to load a watermark image into a "slot".

Using the Memo.Command 25 it is possible to select during the OnNotify event a watermark of a certain slot.

If paramA is 0, the method will return the page number which is about to be painted.

If paramA=1, the slot with the number paramB will be selected.

Example:

Load the watermarks as metafiles using Command(143):

```
private void button5_Click(object sender, System.EventArgs e)
{
        // Fill 3 slots and activate the display
        wpdllInt1.Command( 143, 0, @"c:\1.emf");
        wpdllInt1.Command( 143, 1, @"c:\2.emf");
        wpdllInt1.Command( 143, 2, @"c:\5.emf");
        wpdllInt1.ProcessWPA("ShowWatermark", "1");
}
```

Select the watermark in the OnNotify Event:

```
private void wpdllInt1_OnNotify(object Sender, int Editor, int MsgID)
{
        if (MsgID==29)
        {
              int pagenr =    wpdllInt1.Memo.TextCommand( 25, 0, 0);
                 if ( pagenr==0) // Page 1, select slot 0
                             wpdllInt1.Memo.TextCommand( 25, 1, 0);
                 else if ( pagenr==4) // Page 5, select slot 2
                             wpdllInt1.Memo.TextCommand( 25, 1, 2);
                 else  // otherwise select slot 1
                             wpdllInt1.Memo.TextCommand( 25, 1, 1);

        }
}
```

? Command 26: Modify MergeText Operation

It is also possible to use bookmarks and hyperlinks as merge fields with Memo.MergeText.
This makes it possible to perform nested merge operations.

To do so, please execute Memo.TextCommand(26, 2, 0) to merge in hyperlinks,
Memo.TextCommand(26, 3, 0) to merge into bookmarks
and Memo.TextCommand(26, 1, 0) to use merge fields (default).

? Command 27: read text width and text height

Using this command it is possible to read different values.

if paramA=0 it will read the text height in twips

if paramA=1 it will read the text width in twips

if paramA=2 it will read the height of all pages twips

? Command 28: get page number as displayed_2

The command returns the current page number as it would be displayed by a page number object.

The page numbers start with 1 and respects page numbering which was restarted in a section.

? Command 29: check for elements in the text

The command returns a bitfield with this bits:

1: Tables are used in document
2: Simple numbers are used
3: Numberstyles are used
4: Number levels are used
5: Styles are used

? Command 30: Input the current engine version

This will add the name of the module, its version and if it is 32 or 64 bit to the text.

TextCommand(30,0) will just add the engine DLL name, TextCommand(30,1) will add the complete path.

? Command 31: Add "lore ipsum" text

Command 31 will append random text, i.e.

```
Lorem ipsum perspiciatis voluptatem quae voluptatem totam consequuntur
ipsa consequuntur beatae quia
ipsa explicabo error sit consequuntur lorem sit voluptatem beatae sed
eos aut ab odit sit sit magni dolores
iste odit lorem.
Lorem ipsum eos architecto aut omnis perspiciatis ipsum sunt lorem
beatae totam aperiam, consequuntur qui
omnis ipsum eos ipsa.
Lorem ipsum inventore ratione dicta laudantium, veritatis magni ut
omnis ipsam natus veritatis totam ipsum
ipsum unde quia ipsa accusantium ut aspernatur nemo ipsa.
Lorem ipsum accusantium aperiam, sed ut aspernatur ....
```

If the paramA=0 it will create between 1 and 10 paragraph, otherwise it will create paramA paragraps.
The count of words in each paragraph will be between 5 and 5+random(paramB) words. (The default for paramB is 30).

Example:

```
Memo.CurrPar.AppendNext();
for (int i = 0; i < 30; i++)
{
```

```
            Memo.TextCommand(31, 1);
            Memo.CurrPar.ParASet((int)WPAT.CharFontSize, (i + 1) * 200);
            Memo.CurrPar.AppendNext();
        }
```

4.1.3.2 BK) TextCommandStr

Declaration

```
string TextCommandStr([In] int ComID, [In] int param, [In] string StrParam);
```

Description

This method has been reserved for custom enhancements.
(We will extend the regular API to include some of this enhancements from time to time.)

? Command ID 1 - default paragraph style

Retrieve or set the current default paragraph style. If always returns the name of the current default style. You may pass the name of a style to select a different style to be default. In case bit 2 in param is not set, the style will be created if it does not exist. If you pass an empty string and pass param=1 no style will be the default style.

? Command ID 2 - convert to ANSI

Converts the complete text into ANSI format. A CR-NL pair is added after each line inside a paragraph or cell.
The **codepage** for the conversion can be provided a first parameter.

Hint: Use command 41 to convert the text into unicode.

? Command ID 3 -& 4 - clipboard format

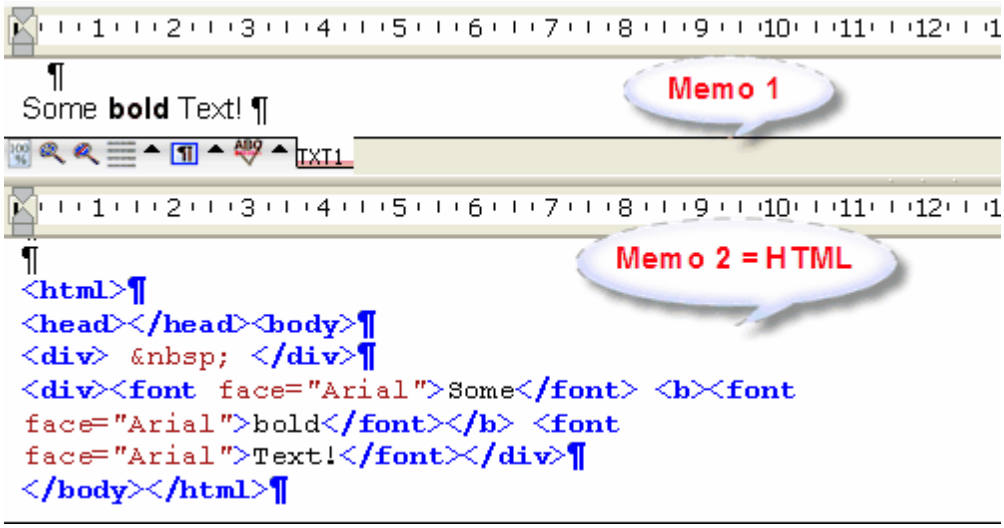
ID 3:
Set the property **TextSaveFormatClipboard**. This property holds the format string which is used to write the text which is copied to the clipboard. It returns the previous value of this property. If param<0 the property is not changed.

ID 4:
Set the property **TextLoadFormatClipboard**. This property holds the format string which is used to read the text which is copied from the clipboard. It returns the previous value of this property. If param<0 the property is not changed.

? Command ID 5 - view source mode

Activates the "view source" mode. Here the double editor mode is used to display the created file including formatting codes in the second editor. If param=1 changes in the the second editor automatically are applied to first editor. You can use this mode with the internal HTML highlighter!

```
wpdllInt1.SetEditorMode(EditorMode.wpmodDoubleEditor);
wpdllInt1.Memo.LayoutMode = LayoutMode.wplayNormal;
wpdllInt1.Memo2.LayoutMode = LayoutMode.wplayNormal;
wpdllInt1.Memo.TextCommandStr(5,1,"HTML-IgnorePageisze");
wpdllInt1.Memo2.TextCommand(12,1,0);
```



To insert markup codes you can use Memo2.TextCursor.InputString or Memo2.TextCommandStr (9,...)

? Command ID 6 - select HTTP DLL

This command activates the HTTP access dll. You need to pass the DLL activation key as string parameter. Using an empty string will unload the DLL. The DLL will be always searched in same directory as the main TextDynamic engine.

The DLL name is always wphttpget.dll, or with TextDynamic demo: wphttpget_demo.dll.

The HTTP DLL can be licensed

€42 for Single License
http://shareit1.element-5.de/cart.html?PRODUCT[300326575]=1

€105 for Site / Team License
http://shareit1.element-5.de/cart.html?PRODUCT[300326576]=1

Once the DLL is loaded all referenced CSS and IMG data will be loaded through the methods exported by the DLL.

Example:

```
wpdllInt1.Memo.TextCommandStr(6, 1, "HTTP_DEMO_MODE");
wpdllInt1.Memo.LoadFromFile(@"http://www.wpcubed.com/");
```

Note: You should also activate a new HTML formatting method to format tables differently
see Memo.SetBProp Group 7 : wpAsWebPage

? Command ID 7 & 8 - set ParIsOutline flag

ID 7:
Processes all paragraphs of the body text to select paragraphs to be used as PDF bookmark (outline).

StrParam="" - disables all WPAT.ParIsOutline property.

If the **paragraph uses the style** with the name provided as StrParam set the ParIsOutline level provided as Param. If Param = 0 or if the paragraph does not contain any text (objects are ignored) the property will be set to 0.

You can set the WPAT.ParIsOutline property in paragraph styles as well.

This command is a shortcut for the following C# code with the advantage that it does not modify the cursor position.

```
IWPTextCursor Cursor = Memo.TextCursor;
IWPParInterface CurrPar= Memo.CurrPar;
Cursor.CPPosition = 0;
bool next = true;
while (next)
{
         // Check for style =   "heading 1"
         if((CurrPar.StyleName=="heading 1") &&
         // Check if this paragraph contains any text (spaces and all objects
                  (CurrPar.ParCommand(1,1,0xFFFF)!=1))
                  CurrPar.ParASet((int)WPAT.ParIsOutline,1);
         else CurrPar.ParADel((int)WPAT.ParIsOutline);
         next = Cursor.CPMoveNextPar(false);
}
```

Note: Use TextCommand(3) to create a TOC.

ID 8:
Processes all paragraphs of the body text to select paragraphs to be used as PDF bookmark (outline).

StrParam="" - disables all WPAT.ParIsOutline property.

If the **paragraph starts with** StrParam set the ParIsOutline level provided as Param. If Param = 0 the property will be set to 0.

? Command ID 9 - insert HTML markup

This command inserts HTML open and closing markup.

`Memo.TextCommandStr(9, 0, "b") creates` **|**.

If text is currently selected the tags will be created around the selected text, if no text is selected the cursor **|** will be placed within the tags.

You can also specify parameters:

Memo.TextCommandStr(9, 0, "span style=\"font-face:Arial\"") creates **|**.

? Command ID 10 - Select Printer

This command can be used to select a different printer by specifying a name.

First the printer is located by looking for an exact match of the name.

Then a printer is located which name starts with the provided string.

At last the printer is located which name contains the passed string.

If no printer was not found the result is 0, otherwise 1.

You can use Command ID 12 to retrieve a list of printer names.

? Command ID 11 - Print Text

Print the current text.

The following string parameter are supported:

@@ODD@@ - prints pages 1,3,5,7....
@@EVEN@@ - prints pages 2,4,6,8, ...

or a page list can be provided. If an empty string was passed the printing mode will be set according to property PrintParameter.

? Command ID 12 - Get Printernames

Return a list of printer names.

***

? Command ID 13 - BeginPrint

Start the printing cue -
this does the same as the method WPDllInt BeginPrint
EndPrint

Also see: Command ID 34 -Print To File

? Command ID 14 - EndPrint

End the printing cue -
this does the same as the method WPDllInt EndPrint

? Command ID 15 - Update ShowFields Mode

This commands hides multiple paragraphs within texts in case the property ShowFields = true.

Example: INTERPRET

Category
Mailmerge

? Command ID 16 - Syntax Highlighting

TextDynamic includes a versatile syntax highlighting interface.

This feature is now controlled by the API ActivateSyntaxHighlighter

Category
Mailmerge

? Command ID 17 - Token To Template Conversion

The token to template conversion uses special character combinations to separate the commands from the text. Fields have to be embedded into the characters **<<** and **>>** (can be customized)

The command ID 17 starts the conversion. The result is "" or an error description.

This conversion can be used to create mail merge documents and report templates. The creation of a report template requires that the reporting has been activated using SetEditorMode.

Please see the report template syntax description!

Parameters

a) Using the integer parameter it is possible to only select certain sub passes of the conversion:

If the parameter is 0 and reporting has been activated a report template will be created. If reporting is not active only merge fields will be created.

b) The string parameter has to be empty or consists of 2 or 4 lines (separated by CRNL)

1. Line = Start code, default <<
2. Line = End code, default >>
3. Line = Band character, default :
4. Line = Group character, default #

Examples

a) Create merge template and merge text:

```
wpdllInt1.Memo.TextCommandStr(17, 3, "");
wpdllInt1.Memo.MergeText("");
```

b) Create reporting template:

```
wpdllInt1.SetEditorMode(
   EditorMode.wpmodDoubleEditor,
   EditorXMode.wpmodexToolbar|
   EditorXMode.wpmodexPDFExport|
   EditorXMode.wpmodexSpellcheck|
   EditorXMode.wpmodexTables|
   EditorXMode.wpmodexReporting,
   EditorGUI.wpguiRuler|
   EditorGUI.wpguiHorzScrollBar|
   EditorGUI.wpguiVertScrollBar|
   EditorGUI.wpguiPanelH1,
   EditorGUI.wpguiDontSet);
wpdllInt1.Memo.TextCommandStr(17, 0, "");
```

Category
Mailmerge

? Command ID 18 - Set default font and size

This command sets the default font name and size. It only needs to be called once.

The font will be applied the next time the editor is cleared.

To disable the default font pass -1 as integer parameter.

Example VB:

```
Private Sub Command4_Click()
  WPDLLInt1.Memo.TextCommandStr 18, 22, "Courier New"
End Sub
```

? Command ID 19 - Make the current page size the default

This command makes the current page size the page size which will be used for new documents.

```
// Set PageSize (use smaller margins, 720 twips)
   wpdllInt1.Memo.PageSize.SetPageWH(-1, -1, 720, 720, 720, 720);

// This should be also used for new documents
   wpdllInt1.Memo.TextCommandStr(19, 0, "");
```

? Command ID 20 - Insert Annotation Object

This command will insert an "annotation object".

The string parameter will be used as Caption.

Annotation objects are painted as little yellow square.

Note: Annotation Objects are currently only used with special XML import/export.

? Command IDs 21-27 Export Pages as BMP, PNG and TIFF.

New:

We added the new API Memo.GetPageAsBitmap which handles this functionality.

So this commands are obsolete.

TextDynamic is able to export certain pages into bitmap files. With the premium or server license it even allows the export into a multipage TIFF file!

Note: Please call Memo.ReformatAll(false,false) if the editor was dynamically created.

Commands to control the bitmap export:

*21: Set the export resolution to the value of param. The default is **200 dpi**. Please avoid too large values if you war exporting color images. The memory consumption of the working bitmap is huge. (The string parameter is ignored)*

If you need different resolutions for X and Y values, please multiply the Y resolution with 10000 and add it:

```
wpdllInt1.Memo.TextCommandStr(24, XResolution + YResolution * 10000, "");
```

22: Select the bit count. param=1 select monochrome (=default), param=2 selects 24 bit color. (The string parameter is ignored)

Note: when exporting monochrome images colored text will be rendered black, shading will be ignored unless dithering mode is selected.

The string parameter can be used to select additional options. Please pass a number, for example "1024". It will be interpreted as bitfield
4: draw a page frame
8: show the selection
32: do not display watermarks
64: don't print header and footer
128: don't export images
256: display gridlines for tables
512: export metafiles as bitmaps
1024: for monochrom bitmaps: select dither mode

23: Export one page to a BMP file. param=page number (0..), the string param is the filename

24: Export one page to a PNG file. param=page number (0..), the string param is the filename

```
private void button1_Click1(object sender, System.EventArgs e)
{
   wpdllInt1.Memo.TextCommandStr(24, 0, @"c:\Page_one.png");
}
```

TIFF Export:

TextDynamic: requires "Premium" or "Server" License! *wRTF2PDF TextDynamic Server: requires "Server" License!*

25: Start a new TIFF file - the string parameter contains the file name. Result = "OK" if successful, otherwise an error message.

26: Closes the current TIFF file. If successful, the page count is returned.

27: Exports one page in the current TIFF file. If successful "OK" is returned. Otherwise the result string is empty or an error message. It uses the parameters defined with the commands 21 and 22. By default the resolution is 200 dpi, monochrome.

This example exports the complete document into a single TIFF file.

```
private void button1_Click1(object sender, System.EventArgs e)
{
      if ( wpdllInt1.Memo.TextCommandStr(25, 0, @"c:\multipage.tif")=="OK" )
      {
               for(int i=0;
                   wpdllInt1.Memo.TextCommandStr(27, i, "")=="OK";
                   i++);

                MessageBox.Show(
                   wpdllInt1.Memo.TextCommandStr(26, 0, "") + " page(s) were export
      };
}
```

Note: It is possible to load a different document and export pages into the already opened TIFF file.

? Command ID 28/29/30/31 - Load & Save the paragraph styles

Command 28 will save the paragraph styles to a file

Command 29 will load the styles

Command 30 will load the CSS data from a string directly

Command 31 will save the CSS data to a string

Also see SaveStyleSheet and LoadStyleSheet (replaces this commands 28 and 29).

Also see EnumParStyles

? Command ID 32 - read character atr cursor position

This command will return a single character at cursor position.
At the end of a paragraph Char(13) will be returned.

You can also use CurrPar to extract text.

? Command ID 33 - read and write EventButton properties

The event OnButtonClick is triggered when the user clicks on a button or menu.
It allows to execute custom actions.

We provide the TextCommandStr(33,...) method to allow the use of custom buttons and menus also in development systems, which cannot access the EventButton interface. **In .NET or VB6 there is no need to use this method.**

Using param You can read the following parameters of the IWPDllButton Interface.

The following ids can be used. The value can also be negative. In this case the property will be modified to the value of StrParam.

1: Name
2: Hint
3: Caption
4: Param (a string)
5: Font
6: Action (a string)
7: WPActionStyle (a number converted to string)
8: Selected (a boolean, "1" or "0")
9: Disabled (a boolean, "1" or "0")
10: ShowCaption (a boolean, "1" or "0")
11: Visible (a boolean, "1" or "0")
12: Typ (a number converted to string)
13: IParam (a number converted to string)

Use param=99 to call ItemsClear() and param>=100 to add ItemsAdd(StrParam, param-100)

? Command ID 34 -Print To File

Set the name of a print file. The subsequent printing will use this file.

? Command ID 35/36/37 change and read lines

This commands will return text or, if the integer parameter is <>0 modify text.

ID35: Read or change current paragraph (CPPar)

ID36: Read or change current line (CPLine)

ID37: Read or change current word (CPWord)

? Command ID 38 - define font drop down items

You can use this command to set a predefined list of fonts which are available in the font drop down.

The list is passed as comma separated list, i.e. "\"Arial\",\"Courier New\""

? Command ID 39: Convert Tables to Text

This command converts tables into text.

The following options:

param:
0 - convert all tables
1 - convert current table only

StrParam:
"" - convert the table into tabbed text. Also handles cells with multiple paragraphs and multiple lines.
"TABPAR" - convert the table into tabbed text. Also handles cells with multiple paragraphs
"TAB" - convert into text with tabs
"PAR" - convert into multiple paragraphs
Otherwise the provided text will be used as separator

? Command ID 40 - define which char is a word delimiter

Using the command 40 You can specify wether a character is a word delimiter or not, a non breaking character. Word delimiters are used to apply word wrap and spellcheck.

Memo.TextCommandStr(40, 1, "-+") makes the characters + and - word delimiters

Memo.TextCommandStr(40, 0, "-+") makes the characters + and - non breaking character

The string may only use characters in the ANSI range.

? Command ID 41 - convert to unicode

Converts the complete text into Unciode format. A CR-NL pair is added after each line inside a paragraph or cell.

? Command ID 42 - move cursor to a paragraph with a certain name

You can use this command to move the cursor to a paragraph with a certain name. (Use CurrPar.ParStrCommand with id 4 to set the name)

param is a bitfield:

bit 1:
0 : Search for the name (case insensitive)
1 : Search for a paragraph which contains the text.

bit 2: (=4)
0 : Position the cursor as if the user would have moved it there
1 : Position the cursor using low level assignment. This makes it possible to reach paragraphs which are not accessible otherwise.

bit 3: (=8)
0 : Return "1" if the cursor was moved, "", if not.
1 : Return the text in the found paragraph.

? Command ID 43-46: Delete, Read and Write a field

ID 43:
This command let you delete a certain merge field with the name *StrParam*.

If param>0 the text of the field will be deleted, otherwise only the start and end markers.

You can pass "*ALL*" as name to delete all fields.

ID 44:
Set the field save and load format, i.e. "RTF", "HTML-softlinebreaks" ... for the next use of ID 45 and ID 46.

ID 45:
Read the embedded text in the format set with ID 44
Param is the object type in the low byte. Use 1 for merge fields.
Add 8192 to Param to search from cursor position, add 4096 to compare the "Command" parameter instead of name.
Add 16384 to globally search for the field.
StrParam is the name of the object which should be saved.

ID 46:
load the embedded text in the format set with ID 44
Param is the object type in the low byte. Use 1 for merge fields.
Add 8192 to Param to search from cursor position, add 4096 to compare the "Command" parameter instead of name.
Add 16384 to globally search for the field.
StrParam is the the fieldname + "=" + new text.

An alternative is to use the regular save and load methods (LoadFromFile, SaveToFile) and specify the fieldname in the format string, i.e.
"f:name=RTF" to save or load the contents of the field "name".

? Command ID 47: Get list of printer tray names

This command reads the list of trays of the current printer. It creates a list in the form "ID1=Name","ID2=Name"

Also see: IWPPrintParameter

4.1.3.2 BL) SelectFirstParGlobal

Declaration
```
bool SelectFirstParGlobal()
```

Description
This method let the interfaces CurrPar and CurParAttr use the globally first paragraph in this document. This paragraph can be also located in a header or footer. You will use this method only if you have to create a loop over all paragraphs in this document.

When your code has been completed please make sure you move the cursor to a defined position, ie use TextCursor.CPPosition = 0;

4.1.3.2 BM) PrepareAttachmentList

Declaration
```
function PrepareAttachmentList(Mode: Integer; const Path: WideString): Integer;
```

Description

When You want to save to HTML You can activate the automatic creation of file attachments for embedded images.

Use Mode=0 to disable this mode.

Use mode=1 to enable this mode and also use the parameter path to set the directory, where the files should be stored.

Use mode=2 to get the count of attached images

You can use GetAttachment to read each single name.

Use mode=3 to delete all files which are listed in the attachment list.

The provided path (directory) will be used to save the images.
It is not necessary to use the format string imgpath:"..." in the save method unless You want to override the path.

4.1.3.2 BN) GetAttachment

Declaration
```
function GetAttachment(Index: Integer): WideString;
```

Description

Reads the name of an attached file. See PrepareAttachmentList.

4.1.3.2 BO) SetInitialPath

Declaration
```
procedure SetInitialPath(Id: Integer; const Value: WideString)
```

Description

Sets various initial directories for file open dialogs:

Use id to select

0 : directory for text load and save. This is the same as property InitialDir
1 : directory for the insert graphic dialog
2 : directory for loading and saving styles sheets

4.1.3.2 BP) GetInitialPath

Declaration

```
function GetInitialPath(Id: Integer): WideString;
```

Description

Reads various initial directories for file open dialogs:

Use id to select

0 : directory for text load and save. This is the same as property InitialDir
1 : directory for the insert graphic dialog.
2 : directory for loading and saving styles sheets

4.1.3.2 BQ) ActivateSyntaxHighlighter

Declaration

```
int ActivateSyntaxHighlighter(int Mode; string param)
```

Description

TextDynamic includes a versatile syntax highlighting interface.

It internally has the possibility to change the attributes of the text (i.e. apply color green for comments) or just add virtual marks. In any case the visible attributes will be updated as the user types.

Currently available is syntax highlighting for XML, for fields and for field and band tokens.

`Memo.ActivateSyntaxHighlighter( param, strparam)` selects the internal Syntax Highlighting Mode.

param selects the mode:

0 - disables the highlighting,

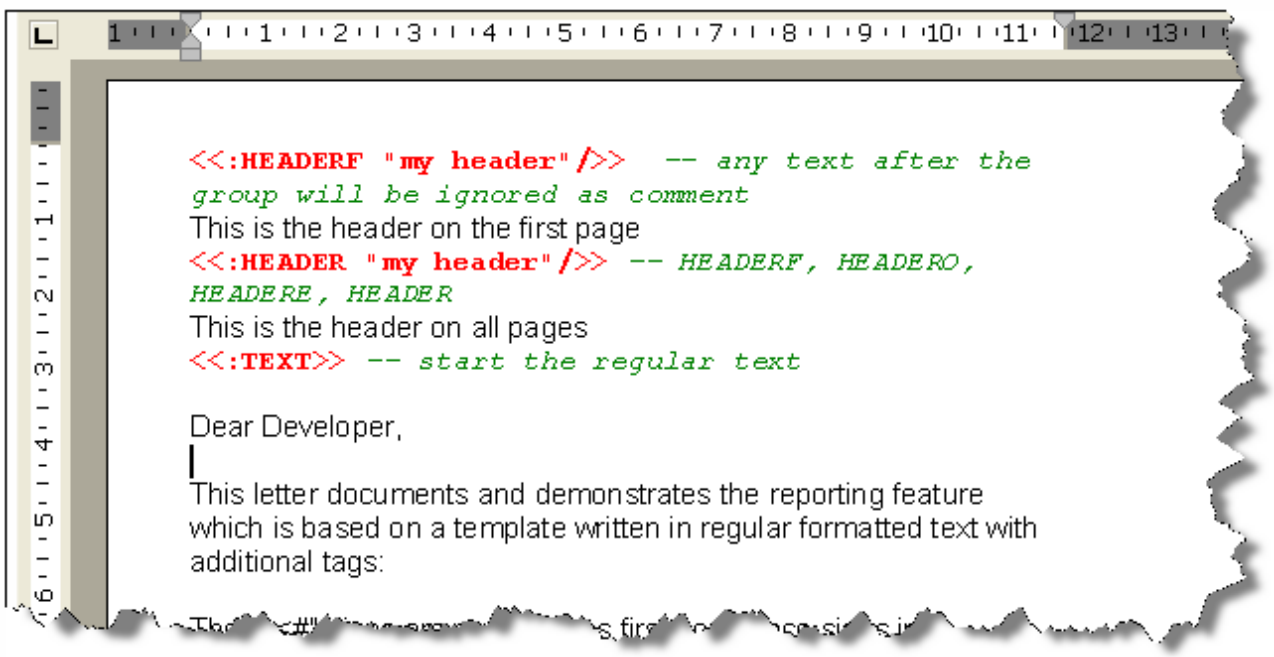
1 - selects the HTML / XML highlighter. (Attributes are applied to the text)

2 - selects the HTML / XML highlighter. (Attributes are virtual, they are not applied to the text)

3 - highlight fields using the syntax <<...>> - see Mailmerge (Attributes are not applied)

4 - highlight fields and bands using the syntax <<...>> - see "Tokens to Template Conversion " (Attributes are not applied). To actually convert the tokes use TextCommandStr(17).

Example:



Modes 3 and 4 can be customized by passing a string parameter.

The string parameter has to consists of 2 or 4 lines (separated by CRNL)
1. Line = Startcode, default <<
2. Line = Endcode, default >>
3. Line = Bandcharacter, default :
4. Line = Groupcharacter, default #

Category
Mailmerge

4.1.3.2 BR) SetXMLSchema

Declaration
```
int SetXMLSchema(string Value; int Mode)
```

Description

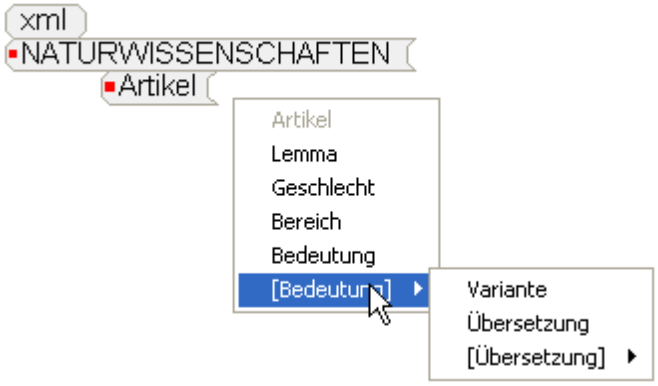
This feature is used to manage an **XML schema** which is used to create popup menus (Mode=0-5) and to load a **CSS stylesheet** to highlight the text in certain tags (Mode=6 + 7).

Note: This method requires the "Premium" License. Since it is an editing feature, You can not use it in TextDynamic *Server*.

You can load an XML schema file, i.e:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" att
   <xs:element name="NATURWISSENSCHAFTEN">
      <xs:annotation>
         <xs:documentation>....</xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Artikel" maxOccurs="unbounded">
            ....
```

This information will be used to create a popup menu when the user clicks inside the text. The User can now select a tag fro



In the present version the control does not check rules in the schema, such as maxOccurs.

? A) Manage XML Schema

Mode=0: Set the schema mode. The first parameter can have this values:
"clear" - clear schema and disable the XML editing
"off" - don't clear the schema, but switch editing off
"on" - enable XML editing and XML Popup

Mode=1: Load the schema from the file "Value". Use SetXMLSchema("on",0) to activate XML editing!

Mode=2: Save the schema to a file.

Mode=3: Load the schema from a string.

Mode=4: Load XML data from a file into the editor. Alternatively this can be done using the regular load methods when using

Mode=5: Save XML data in editor to a file. Alternatively this can be done using the regular save methods when using the for

Example:

```
wpdllInt1.Memo.SetXMLSchema(@"\XMLSchema\naturwissenschaften.xml", 4);
wpdllInt1.Memo.SetXMLSchema(@"\XMLSchema\naturwissenschaften.xsd", 1);
wpdllInt1.Memo.SetXMLSchema("on", 0);
```

? B) Load CSS Stylesheet

Text inside certain styles can be formatted using a style sheet.
If the CSS mode is active, the text inside elements with a name which is equal to a paragraph style uses the font definition o

Mode=6:
Activate the CSS mode: `wpdllInt1.Memo.SetXMLSchema("on", 6);`
Deactivate the CSS mode: `wpdllInt1.Memo.SetXMLSchema("off", 6);`

```
// You also need to execute wpdllInt1.Memo.SetXMLSchema("on", 0);
```

Mode=7: assign the style sheet from a string (CSS syntax)

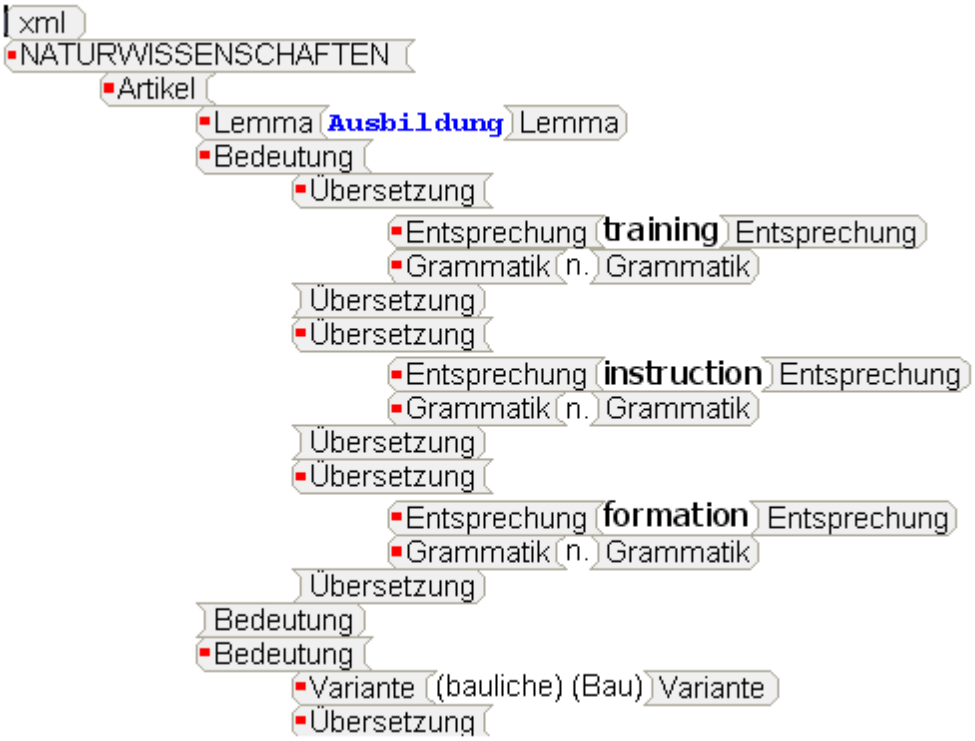
CSS styles can be also loaded into the paragraph style collection using LoadStyleSheet.

Example CSS:

```
Lemma { font-family:'Courier New';font-weight:bold;color:blue; }
```

Entsprechung { font-family:'Tahoma';font-size:13.00pt; }

Screenshot:



? C) Example Code

Example Code C#:

```
wpdllInt1.Memo.SetXMLSchema("some_xml.xml", 4);
wpdllInt1.Memo.SetXMLSchema("some_schema.xsd", 1);
wpdllInt1.Memo.LoadStyleSheet("some_stylesheet.css");
wpdllInt1.Memo.SetXMLSchema("on", 6); // CSS Mode
wpdllInt1.Memo.SetXMLSchema("on", 0); // XML Mode
```

4.1.3.2 BS) GetXMLTags

Declaration
`function GetXMLTags(Mode: Integer): WideString`

Description

The method is reserved

4.1.3.2 BT) ShowContextMenu

Declaration
`function ShowContextMenu(Mode: Integer): Integer`

Description

This method displays the current context menu at the cursor position. It does the same as pressing the "Context Menu" key on the keyboard.

4.1.3.2 BU) InitHTMLMode

Declaration
```
procedure InitHTMLMode(Mode: Integer; const ParamStr: WideString; param: Integer)
```

Description

This method initialites the HTTP

4.2 IWPTextCursor (Text creation and cursor positioning)

Text creation and cursor positioning

Description
The TextCursor interface is used to create text under program control (Input functions) and to position the cursor and select text.

Note: To add tabstops in the selected text please use Memo.TextCommand(21, ...)

Use **Memo.TextCursor** to get a reference to this powerful and important interface.

To move the cursor to the start of the paragraph referenced by CurrPar use Memo.TextCommand(29)

Tip:
To read the character at the current position use
`Memo.CurrPar.GetChar(Memo.TextCursor.CPPosInPar)`

Please note, the text box and foot note support has to be activated using SetEditorMode!

Properties	Methods	Methods	Methods
CPCellPtr	AddTable	GetParName	MarkerCollect
CPLineNr	AppendRow	GotoBody	MarkerCollectAll
CPObjPtr	CheckState	GotoEnd	MarkerCPPosition
CPPageLineNr	Clear	GotoStart	MarkerDrop
CPPageNr	CombineCellHorz	HideSelection	MarkerGoto
CPParNr	CombineCellVert	InputBookmark	MarkerSelect
CPParPtr	CPMoveAfterTable	InputCalculatedField	MovePosition
CPPosInLine	CPMoveBack	InputCell	MoveToField (locates next/ previous field)
CPPosInPar	CPMoveBeforeTable	InputCustomHTML	MoveToBookmark
CPPosition	CPMoveNext	InputEmbeddedData	MoveToTable
CPRowPtr	CPMoveNextBand	InputField	Redo
CPStylePtr	CPMoveNextCell	InputFieldObject	ReplaceText
CPTableColNr	CPMoveNextGroup	InputFooter	ReportConvertTable
CPTablePtr	CPMoveNextObject	InputFootnote	ReportConvertTe
CPTableRowNr	CPMoveNextPar	InputHeader	
IsSelected	CPMoveNextRow	ClearAllHeaders	
PageCount	CPMoveNextTable		

CPMoveParentTable
CPMovePrevCell
CPMovePrevObject
CPMovePrevPar
CPMovePrevRow
CPMovePrevTable
CPMoveToGroup
CPOpenObj
EnumOpenObj
Delete
DisableProtection
DisableUndo
EnabledProtection
EnableUndo
FieldsFromTokens
FindText
ClearAllFooters
InputHTML
InputHyperlink
InputImage
InputObject
InputPagebreak
InputParagraph
InputPicture
InputPictureStream
InputRowEnd
InputRowStart
InputSection
InputString
InputTable
InputTabstop
InputText
InputTextbox
InsertRow
ExitTable
TableSplit
TableSort
ASetCellProp
ASetCellStyle
MergeCellHorz
MergeCellVert
xt
ReportInputBand
ReportInputGroup
ScrollToCP
SelectAll
SelectLine
SelectParagraph
SelectTable
SelectTableColumn
SelectTableRow
SelectText
SetColWidth
SetParName
SetRowHeight
SetTableLeftRight
TableClear
TableDelete
Undo
UndoClear
WordCharAttr
WordEnum
WordHighlight

4.2.1 Properties

4.2.1.1 CPCellPtr

Applies to
IWPTextCursor

Declaration
```
int CPCellPtr;
```

Description
This is a low level paragraph reference to the cell the cursor is located within.

4.2.1.2 CPLineNr

Applies to
IWPTextCursor

Declaration
```
int CPLineNr;
```

Description
This is the number of the current line from start of the text.

Category
Display Status Information

Also see
Memo.TextCommandStr ID 35/36/37 to change and read paragraphs, lines and words at cursor position.

4.2.1.3 CPObjPtr

Applies to
IWPTextCursor

Declaration
```
int CPObjPtr;
```

Description
This is a low level text object reference to the object at cursor position.

4.2.1.4 CPPageLineNr

Applies to
IWPTextCursor

Declaration
```
int CPPageLineNr;
```

Description
This is the number of the current line from start of the page.

4.2.1.5 CPPageNr

Applies to
IWPTextCursor

Declaration
```
int CPPageNr;
```

Description
This is the number of the current page.

Also see TextCommand(28)

Category
Display Status Information

4.2.1.6 CPParNr

Applies to
IWPTextCursor

Declaration
```
int CPParNr;
```

Description
This is the number of the current paragraph.

4.2.1.7 CPParPtr

Applies to
IWPTextCursor

Declaration
```
int CPParPtr;
```

Description
This is a low level paragraph reference to the paragraph the cursor is located within.

4.2.1.8 CPPosInLine

Applies to
IWPTextCursor

Declaration
```
int CPPosInLine;
```

Description
This is the position in the current line.

Category
Display Status Information

4.2.1.9 CPPosInPar

Applies to
IWPTextCursor

Declaration
```
int CPPosInPar;
```

Description
This is the position in the current paragraph.

4.2.1.10 CPPosition

Applies to
IWPTextCursor

Declaration
```
int CPPosition;
```

Description
This is the absolute position in characters from the beginning of the text. Paragraph breaks count as 1.

Category
Display Status Information

4.2.1.11 CPRowPtr

Applies to
IWPTextCursor

Declaration
```
int CPRowPtr;
```

Description
This is a low level paragraph reference to the row the cursor is located within.

4.2.1.12 CPStylePtr

Applies to
IWPTextCursor

Declaration
```
int CPStylePtr;
```

Description
This is the number of the style the current paragraph uses. You can assign 0 to clear the style reference.

4.2.1.13 CPTableColNr

Applies to
IWPTextCursor

Declaration
```
int CPTableColNr;
```

Description
This is the number of the current column.

Category
Table Support

4.2.1.14 CPTablePtr

Applies to
IWPTextCursor

Declaration
```
int CPTablePtr;
```

Description
This is a low level paragraph reference to the table the cursor is located within.

Category
Table Support

4.2.1.15 CPTableRowNr

Applies to
IWPTextCursor

Declaration
```
int CPTableRowNr;
```

Description
This is the nbumber of the current row. Y

Category
Table Support

4.2.1.16 IsSelected

Applies to
IWPTextCursor

Declaration
```
bool IsSelected;
```

Description
This property is true if text is selected.

4.2.1.17 PageCount

Applies to
IWPTextCursor

Declaration
```
int PageCount;
```

Description
The count of pages in the document - readonly.

Also see category "Display Status Information"

4.2.1.18 SelStart

This is the start position of the selected text.

To **move** the cursor please better use IWPTextCursor.MovePosition or CPPosition.

To **select** the text use IWPTextCursor.SelectText

4.2.1.19 SelLength

This is the length of the selected text. Also see SelStart.

4.2.2 Methods

4.2.2.1 IWPTextCursor.AddTable

Create a table and use callback event

Applies to
IWPTextCursor

Declaration
```
procedure AddTable(const TableName: WideString; ColCount: Integer; RowCount:
Integer; Border: WordBool; EventParam: Integer; CreateHeader: WordBool;
CreateFooter: WordBool);
```

Description

This method creates a table and triggers the callback OnCreateNewCell for each created cell if EventParam was passed with a value != 0. So you can add the data and properties is a very efficient way.

Since the cursor will be placed inside the first cell, you can, in case it is not possible to use the event, fill text in each new cell by moving the cursor through the table using CPMoveNextCell and CPMoveNextRow. **To create text after the table use ExitTable or InputParagraph (2,"") !**

Parameters	
Name	An optional name for the table. You can use MoveToTable to later locate the table.
ColCount	The count of columns which should be created
RowCount	The count of rows which should be created (not including header/footer rows). You can abort the creation in event OnCreateNewCell by changing the value of the boolen variable AbortAtRowEnd.
Border	Create a broder around cells. You can also set border attributes in event OnCreateNewCell.
EventParam	Parameter passed through to OnCreateNewCell. This event is not called if this parameter is 0.
CreateHeader	If true an extra header row with rownr=-1 will be created. The formatting routine is able to repeat header rows at the start of a page.
CreateFooter	If true an extra footer row with rownr=-2 will be created. The formatting routine is able to repeat footer rows at the start of a page. (This special feature is not supported by MSWord)

Tip: You can use the methods ASetCellProp and ASetCellStyle to quickly modify the properties, text and styles of certain cells in an existing table.

Category
Callback Functions
Table Support

4.2.2.2 IWPTextCursor.AppendRow

Applies to
IWPTextCursor

Declaration
```
bool AppendRow(string TableName, int FooterCount);
```

Description
This method will append a row to a table. It is the prefect tool to modify a calculated table in an invoice since it can reserve a variable count of rows at the bottom of the table as footer rows.
If operation was successful (result=true) the cursor will be positioned in the first cell of the new row.

Parameters	
TableName	If specified the table with this name will be searched and extended.

	If this name = "@@COPYTEXT@@" the last row of the current table will be duplicated including contained text.
FooterCount	Count of rows which should be used as footer table rows. The method will be inserted before the footer area.

The text in the row is not duplicated unless the mode DuplicateWithText has been activated using Memo.SetBProp(18,1);

Category
Table Support

4.2.2.3 IWPTextCursor.CheckState

Applies to
IWPTextCursor

Declaration
```
function CheckState(Which: Integer): WordBool;
```

Description
Checks several states:

WPSTAT_ISSELECTED = 0;
Return true if currently text is selected.

WPSTAT_INTABLE = 1;
Return true if the cursor is currently inside a table.

WPSTAT_ISEMPTY = 2;
Return true if the text is empty.

WPSTAT_ISFIRSTLINE = 3;
Return true if the cursor is in the first line of the text.

WPSTAT_ISLASTLINE = 4;
Return true if the cursor is in the last line of the text.

WPSTAT_CANUNDO = 5;
Return true if UNDO is possible.

WPSTAT_CANREDU = 6;
Return true if REDO is possible.

WPSTAT_CANEDIT = 7;
Return true if the text is editable. This also triggers the PropRequestEdit event.

WPSTAT_MODIFIED = 8;
Return true if the text was modified.

WPSTAT_CLEARMODIFIED = 9;
Return true if the text was modified and clears this flag.

WPSTAT_CHANGEAPPLIED = 10;

Return true if the text was modified and **triggers the PropChanged event to make sure programmatic changes in the text are saved.**
In a data bound editor this can cause a reload of the text - so please call this method at first chance before any code which updates the text.

WPSTAT_ISSELECTEDINTABLE = 11;
Return true if currently table cells are selected.

WPSTAT_UpdateUndoState = 12;
Triggers the OnUpdateGUI event to report a change in the undo stack.

WPSTAT_DoUpdateCharAttr=13;
Triggers the OnUpdateGUI event to report a change of the character attributes.

WPSTAT_DoUpdateParAttr=14;
Triggers the OnUpdateGUI event to report a change of the paragraph attributes.

4.2.2.4 IWPTextCursor.Clear

Applies to
IWPTextCursor

Declaration
```
procedure Clear;
```

Description
Clears the current text layer (data block) - does not clear number- or paragraph styles.

4.2.2.5 IWPTextCursor.CombineCellHorz

Applies to
IWPTextCursor

Declaration
```
function CombineCellHorz: WordBool;
```

Description
Merges the selected cells horizontally. Also see IWPTextCursor.MergeCellHorz.

Category
Table Support

4.2.2.6 IWPTextCursor.CombineCellVert

Applies to
IWPTextCursor

Declaration
```
function CombineCellVert: WordBool;
```

Description
Merges the selected cells vertically. Also see IWPTextCursor.MergeCellVert.

Category
Table Support

4.2.2.7 IWPTextCursor.CPMoveAfterTable

Applies to
IWPTextCursor

Declaration
```
function CPMoveAfterTable(CreateIfNotExist: WordBool): WordBool;
```

Description
Moves the cursor at first regular paragraph after the table(s) at the current position. If CreateIfNotExist = true a new paragraph will be created if there is no regular paragraph.

Category
Table Support

4.2.2.8 IWPTextCursor.CPMoveBack

Applies to
IWPTextCursor

Declaration
```
function CPMoveBack: WordBool;
```

Description
Moves the cursor back one character. If this was successful return true, otherwise false.

4.2.2.9 IWPTextCursor.CPMoveBeforeTable

Applies to
IWPTextCursor

Declaration
```
function CPMoveBeforeTable(CreateIfNotExist: WordBool): WordBool;
```

Description
Skips table(s) at current position. Optionally create a new regular paragraph at the beginning of the text.

Category
Table Support

4.2.2.10 IWPTextCursor.CPMoveNext

Applies to
IWPTextCursor

Declaration
```
function CPMoveNext: WordBool;
```

Description
Moves the cursor one character to the right. If this was successful return true, otherwise false.

4.2.2.11 IWPTextCursor.CPMoveNextBand

Applies to
IWPTextCursor

Declaration
```
function CPMoveNextBand: WordBool;
```

Description
__Reserved__

4.2.2.12 IWPTextCursor.CPMoveNextCell

Applies to
IWPTextCursor

Declaration
```
function CPMoveNextCell: WordBool;
```

Description
Move to the cell to the right. If in last cell return false.

Category
Table Support

4.2.2.13 IWPTextCursor.CPMoveNextGroup

Applies to
IWPTextCursor

Declaration
```
function CPMoveNextGroup: WordBool;
```

Description
__Reserved__

4.2.2.14 IWPTextCursor.CPMoveNextObject

Applies to
IWPTextCursor

Declaration
```
function CPMoveNextObject(ObjType: TxTextObjTypes; DontSkipCloseTags: WordBool):
WordBool;
```

Description
Locate next object of a given object type (ObjType).

It will skip closing tags unless the second parameter is true.

You can access the properties of the object or field using CurrObj.

Please also see Memo.EnumTextObj if you need to check all objects in a text.

4.2.2.15 IWPTextCursor.CPMoveNextPar

Applies to
IWPTextCursor

Declaration
```
function CPMoveNextPar(CreateIfNotExists: WordBool): WordBool;
```

Description
Move to next paragraph.

4.2.2.16 IWPTextCursor.CPMoveNextRow

Applies to
IWPTextCursor

Declaration
```
function CPMoveNextRow(CreateIfNotExist: WordBool): WordBool;
```

Description
Move to next row in this table. If the cursor is at the end of a table, optionally a new row can be created by duplicating the current row.

The text in the row is not duplicated unless the mode DuplicateWithText has been activated using Memo.SetBProp(18,1);

Example:

```
IWPMemo Memo;
Memo = WPDLLInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;

// Only one row
TextCursor.AddTable(
           "data", 3, 1, true,
           0, // EventParam =0, no event!
           false,// create header rows
           false // create footer rows
         );
// append rows
for (int r = 0; r < 10; r++)
       {
           // Create new row - if not in first
           if (r > 0) TextCursor.CPMoveNextRow(true);
           // and create the text
           for (int c = 0; c < 3; c++)
           {
               TextCursor.CPTableColNr = c;
               TextCursor.InputText("some text");
           }
       }
TextCursor.InputParagraph(2, ""); //mode=2 --> after table!
Memo.Reformat();
```

If you want to protect cells use this code:

```
IWPParInterface Par = Memo.CurrPar;
Par.ParASet((int)WPAT.ParProtected, 1);
```

Category
Table Support

4.2.2.17 IWPTextCursor.CPMoveNextTable

Applies to
IWPTextCursor

Declaration
```
function CPMoveNextTable: WordBool;
```

Description
Move to the next table after the current.

Category
Table Support

4.2.2.18 IWPTextCursor.CPMoveParentTable

Applies to
IWPTextCursor

Declaration
```
function CPMoveParentTable: WordBool;
```

Description
If in a nested table move to parent table.

Category
Table Support

4.2.2.19 IWPTextCursor.CPMovePrevCell

Applies to
IWPTextCursor

Declaration
```
function CPMovePrevCell: WordBool;
```

Description
Move to the cell to the left. If in last cell return false.

4.2.2.20 IWPTextCursor.CPMovePrevObject

Applies to
IWPTextCursor

Declaration
```
function CPMovePrevObject(ObjType: TxTextObjTypes): WordBool;
```

Description
Locate previous object of a given object type.

4.2.2.21 IWPTextCursor.CPMovePrevPar

Applies to
IWPTextCursor

Declaration

```
function CPMovePrevPar: WordBool;
```

Description
Move to previous paragraph.

4.2.2.22 IWPTextCursor.CPMovePrevRow

Applies to
IWPTextCursor

Declaration
```
function CPMovePrevRow: WordBool;
```

Description
Move to previous row. If at start of table return false.

4.2.2.23 IWPTextCursor.CPMovePrevTable

Applies to
IWPTextCursor

Declaration
```
function CPMovePrevTable: WordBool;
```

Description
Move to previos table before the current table.

4.2.2.24 IWPTextCursor.CPMoveToGroup

Applies to
IWPTextCursor

Declaration
```
function CPMoveToGroup(const Name: WideString; InsideOfCurrentGroup: WordBool;
CreateIfNotExist: WordBool; BandElement: Integer): WordBool;
```

Description
__Reserved__

4.2.2.25 IWPTextCursor.CPOpenObj

Applies to
IWPTextCursor

Declaration
```
function CPOpenObj(ObjTypMask: Integer): IWPTextObj;
```

Description
Checks which objects are open at the current position. Returns a IWPTextObj interface or null if no object is found. Use ObjTypMask=0 it will find any open object, otherwise the specified object type must match.

Note: Please don't forget to call ReleaseInt() with the returned interface at the end of your code.

4.2.2.26 IWPTextCursor.Delete

Applies to
IWPTextCursor

Declaration
```
procedure Delete(CharCount: Integer);
```

Description
Delete "CharCount" characters at the current position. Will also delete paragraph breaks.

4.2.2.27 IWPTextCursor.DisableProtection

Applies to
IWPTextCursor

Declaration
```
procedure DisableProtection;
```

Description
Disable the protection of the text temporarily. Enable the protection with EnableProtection.

4.2.2.28 IWPTextCursor.DisableUndo

Applies to
IWPTextCursor

Declaration
```
procedure DisableUndo;
```

Description
Disable undo temporarily.

Category
Standard Editing Commands

4.2.2.29 IWPTextCursor.EnabledProtection

Applies to
IWPTextCursor

Declaration
```
procedure EnabledProtection;
```

Description
Use after DisableProtection to enable to protect the text again.

Category
Standard Editing Commands

4.2.2.30 IWPTextCursor.EnableUndo

Applies to
IWPTextCursor

Declaration

```
procedure EnableUndo;
```

Description
Enable undo after it was dsiabled with DisableUndo.

Category
Standard Editing Commands

4.2.2.31 IWPTextCursor.FieldsFromTokens

Applies to
IWPTextCursor

Declaration
```
void FieldsFromTokens(string StartText, string EndText, string FieldPreText);
```

Description
Create a merge template by conversion of specially marked fields, i.e [NAME] will be converted into «DB.NAME».

Alternatively You can use the Token To Template Conversion

Please also see Syntax Highlighting since it is possible to highlight fields while the user writes.

Tip: The method FieldsFromTokens does the same as "ReplaceTokens" in the VCL product WPTools.

Parameters

StartText - The field marker start, i.e. "["

EndText - The field marker end, i.e. "]"

FieldPreText - The text which will be added at the beginning of each field, i.e. "DB.".

Category
Mailmerge

4.2.2.32 IWPTextCursor.FindText

Applies to
IWPTextCursor

Declaration
```
void FindText(
   string Text,
   bool FromStart,
   bool CaseSensitive,
   bool MoveCursor,
   bool WholeWords,
   ref bool Found);
```

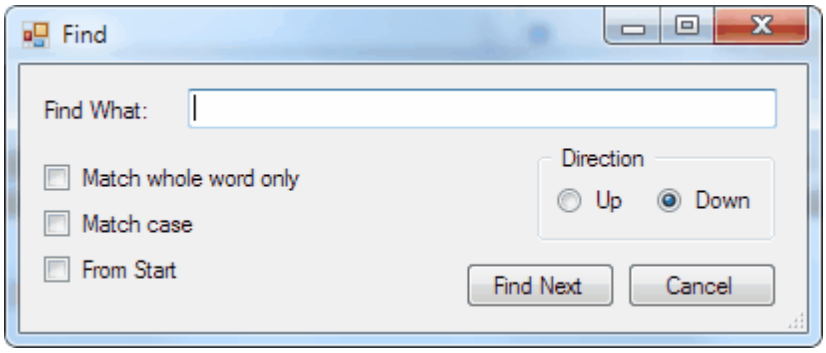
Description

Locates a text and either moves the cursor before its location or selects the text with the cursor being after the found text.

Use Memo.SetIProp(11,-1) to make FindText search backwards.

Parameters	
Text	The text to be searched and selects it or moves the cursor.
FromStart	TRUE to start at the beginning of the text
CaseSensitive	TRUE to search case sensitive
MoveCursor	TRUE to move the cursor to the found position. Otherwise the text will be selected and the cursor will not be moved after the selection. This way a subsequent find operation will locate the next occurrence.
WholeWords	TRUE to find whole words only.
Found	Will be set to TRUE if the text was found.

Example 1 - implement a standard find dialog:



The dialog uses this relevant GUI elements:

```
private System.Windows.Forms.TextBox FindWhat;
private System.Windows.Forms.CheckBox chkWholeWord;
private System.Windows.Forms.CheckBox chkCaseSensitive;
private System.Windows.Forms.RadioButton chkDown;
private System.Windows.Forms.RadioButton chkUp;
private System.Windows.Forms.Button btnFindButton;
private System.Windows.Forms.Button btnCancelButton;
private System.Windows.Forms.CheckBox chkFromStart;
```

This is the constructor:

```
        // Must be assigned before the dialog is displayed!
        IWPMemo Memo;

        public FindDialog(IWPMemo aMemo)
        {
            Memo = aMemo;
            InitializeComponent();
        }
```

This code is executed when the "Find Next" button is pressed:

```
        private void FindButton_Click(object sender, EventArgs e)
        {
            bool found = false;

            // Change Find Direction
            if (chkUp.Checked)
```

```
                Memo.SetIProp(11, -1); // Up=backwards
            else Memo.SetIProp(11, 1); // Down

            Memo.TextCursor.FindText(FindWhat.Text,
                  chkFromStart.Checked,
                  chkCaseSensitive.Checked,
                  false, //if Movecursor=false then select the found text
                  chkWholeWord.Checked, ref found);

            if (found == false)
                MessageBox.Show("Text was not found!");
            else chkFromStart.Checked = false;

        }
```

To use the dialog You can call:

```
 (new FindDialog(wpdllInt1.Memo)).ShowDialog();
```

Example 2 - modify the font of all occurrences of "some text".

```
IWPMemo Memo = wpdllInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;
IWPAttrInterface Attr = Memo.CurrSelAttr;
bool ok = true;

int cp = TextCursor.MarkerDrop(0);
TextCursor.GotoStart();
while(ok)
{
    TextCursor.FindText("some text", false, false, false, false, ref ok);
   if(ok)
   {
      Attr.Clear();
      Attr.SetFontface("Arial");
   }
}
TextCursor.MarkerGoto(true, cp);
```

4.2.2.33 IWPTextCursor.GetParName

Applies to
IWPTextCursor

Declaration

```
string GetParName(int ParagraphPtr);
```

Description
Retrieves the name for a paragraph with the given ID.

ParagraphPtr must be a valid paragraph ID or one of the following constants:

1: Reads the name of the current paragraph

2: Reads the name to the parent table of this paragraph.

3: Reads the name to the parent-parent table of this paragraph.

4: Reads the WPAT_Par_Name of the parent table

5: Reads the WPAT_Par_Name of the parent-parent table

6: Reads the WPAT_Par_Command of the parent table

7: Reads the WPAT_Par_Command of the parent-parent table

4.2.2.34 IWPTextCursor.GotoBody

Applies to
IWPTextCursor

Declaration
```
procedure GotoBody;
```

Description
After the creation of a header, footer, textbox or footnote texts you can use GotoBody() to return to the body text to create more text there.

Category
Hader and Footer Support

4.2.2.35 IWPTextCursor.GotoEnd

Applies to
IWPTextCursor

Declaration
```
procedure GotoEnd;
```

Description
Move cursor to the end of the text.

4.2.2.36 IWPTextCursor.GotoStart

Applies to
IWPTextCursor

Declaration
```
procedure GotoStart;
```

Description
Move cursor to the start of the text.

4.2.2.37 IWPTextCursor.HideSelection

Applies to
IWPTextCursor

Declaration
```
procedure HideSelection;
```

Description

Hide (not clear) the selection.

4.2.2.38 IWPTextCursor.InputBookmark

Applies to
IWPTextCursor

Declaration
```
function InputBookmark(const Name: WideString; const Text: WideString;
CursorWithin: WordBool): Integer;
```

Description
Creates a bookmark. Optionally also create the embedded text and/or places the cursor within the markers.

Category
Hyperlinks and Bookmarks

4.2.2.39 IWPTextCursor.InputCalculatedField

Applies to
IWPTextCursor

Declaration
```
procedure InputCalculatedField(const Command: WideString; const DefaultText:
WideString);
```

Description
Create a text object (NOT a merge field) which is updated using the specified formula.

Parameters	
Command	The formula
DefaultText	The default text displayed by the object

4.2.2.40 IWPTextCursor.InputCell

Applies to
IWPTextCursor

Declaration
```
void InputCell(string Text, sting StyleName);
```

Description
Creates a new Cell in the row which was started with InputRowStart. The borders will be added if bit 1 was set in the mode parameter for method InputRowStart. You can also change the border and other paragraph attributes by accessing the interface CurrPar.

Parameters
Text : Initialization text for this cell. **You can use CurrPar to modify the cell.**
Stylename : Name of the paragraph style this cell should use.

Example:

Create a table with inserted mail merge fields.

```
IWPMemo Memo;
Memo = WPDLLInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;
```

```
IWPAttrInterface CurrParAttr = Memo.CurrParAttr;
IWPParInterface par = Memo.CurrPar;

TextCursor.InputTable(0, "data");

For (int r = 1; r <= 10; r++)
{
    TextCursor.InputRowStart(1);
    if (r & 1 == 1) // odd rows
    {
                            for (int c = 0; c < 3; c++)
                            {
                                TextCursor.InputCell("", "");

                                if (c < 2) par.ParASet((int)WPAT.COLWIDTH_PC, 20 * 100);
                                if (c == 2) par.ParASet((int)WPAT.COLWIDTH_PC, 60 * 100);

                                if (c == 0) TextCursor.InputField("NR_A", "NR_A_TEXT", fa
                                if (c == 1) TextCursor.InputField("NR_B", "NR_A_TEXT", fa
                                if (c == 2) TextCursor.InputField("DESCRIPTION", "DESCRIP
                                par.ParASet((int)WPAT.BorderType, 15);
                                par.ParASet((int)WPAT.BorderWidth, 25);
                            }
     }
     else // even rows
     {
                            for (int c = 0; c < 3; c++)
                            {
                                TextCursor.InputCell("", "");

                                if (c == 0) TextCursor.InputField("MEMO", "MEMO_TEXT", fa
                                else par.IsColMerge = true;

                                par.ParASet((int)WPAT.BorderType, 15);
                                par.ParASet((int)WPAT.BorderWidth, 25);
                                par.ParASet((int)WPAT.BorderWidth, 25);
                            }
      }
      TextCursor.InputRowEnd();
}
TextCursor.InputParagraph(0,"");
Memo.Reformat();
```

Note: In this example we create fields in each cell using InputField. When you merge long texts suing the event OnFieldGetText event you will need Contents.ContinueOptions(256) to make sure the paragraph attributes of the inserted text are preserved.

Category
Table Support

4.2.2.41 IWPTextCursor.InputCustomHTML

Applies to
IWPTextCursor

Declaration

```
procedure InputCustomHTML(const HTMLCode: WideString; const DisplayedText:
WideString);
```

Description
Creates a text object which will export custom HTML code when writing the HTML file.

4.2.2.42 IWPTextCursor.InputEmbeddedData

Applies to
IWPTextCursor

Declaration
int InputEmbeddedData(string imagefilename, int Mode,string datafilename, string dataparams, int datamode);

This method can be used to create a certain object which can also hold binary data. The binary data will be embedded into a PDF file when the internal PDF creation (optional feature) is used.

It is not possible to change the image later using IWPTextObj.LoadFromStream and IWPTextObj.LoadFromFile, these methods will update the attached data, not the image. The method Contents_LoadFromFile however will update the image data.

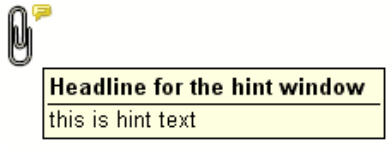
Parameters	
imagefilename	The image file, may be BMP, WMF, EMF, JPG or PNG format. You can also use "pin", "graph", "tag" and "clip" to create an icon. The icon will be only displayed by Acrobat Reader after the export to PDF.
Mode	If bit 1 is set the image will be linked, not embedded. If bit 2 is set the image will be positioned relatively to a paragraph (movable image). If bit 3 is set the image will be positioned relatively to a page (movable image). If bit 4 is set text will not wrap around image (if a movable image). When bit 5 was set and the image was not found an error text object will be inserted. The error object will display <*filename*?>.
datafilename	The file name of the data to be attached
dataparams	Optional parameter list. You can use the parameter names contents and title. **Example:** "\"Contents=this is hint text\",\"Title=Headline for the hint window\""
datamode	should be 0

Returns
0 if not successful. Otherwise the interface CurrObj can be used to change image properties.

C# Example

```
wpdllInt1.Memo.TextCursor.InputEmbeddedData(
   "CLIP",
   0,
   "C:\\my document.rtf",
   "\"Contents=this is hint text\",\"Title=Headline for
   0);
```

Display in AcrobatReader



You can load data from a stream in this object by using CurrObj.LoadFromStream

```
System.IO.Stream str = new System.IO.MemoryStream();
pdf.Memo.SaveToStream(
    new WPDynamic.Stream2WPStream(str) ,false,"RTF");
pdf.Memo.CurrObj.LoadFromStream("Document.RTF",
```

```
    new WPDynamic.Stream2WPStream(str));
```

4.2.2.43 IWPTextCursor.InputField

Applies to
IWPTextCursor

Declaration
```
int InputField(string Name, string Text, bool CursorWithin)
```

Description
To insert a field use is method. It expects the name of the field and the initial text. The boolean parameter controls if the cursor should be placed inside (true) or after the new field (false). Using the *true* you can insert multiple paragraphs or an image inside of the field.

Parameters	
Name	The name of the field
Text	The embedded text - this is initially displayed.
CursorWithin	"True" to place the cursor within the field markers (to add more text or an image), otherwise use "false".

Example:
```
IWPTextCursor TextCursor;
TextCursor = wpdllint1.Memo.TextCursor;
TextCursor.InputField("NAME","This is the name field",false);
```

After you inserted a field you can manipulate the created fieldmarkes using the interface CurrObj. For example you can use Memo.CurrObj.Mode = 2 to convert this field into an "edit field". Such fields mark the only editable text when the editor is in "FormCompletion" Mode.

Please also see Category Mailmerge and the "mail merge API introduction".

To delete, read or write data from a field use: Memo.TextCommandStr ID 43: Delete a field,
to move to a field use IWPTextCursor.MoveToField.

4.2.2.44 IWPTextCursor.InputFieldObject

Applies to
IWPTextCursor

Declaration
```
int InputFieldObject(string Name, string Command, string DefaultText)
```

Description
Creates a text object. This are single objects which are updated by the the event OnTextObjectGetText.

The following field names are predefined:
"PAGE" : The current page number. Parameter "Command" must be empty

"NEXTPAGE": Page number of the next page, empty string if last page
"PRIORPAGE": Previous page number, empty string if first page
"NUMPAGES": Count of pages
"DATE": The current date
"TIME": The current time
"SECTIONPAGES": Count of pages in this section

The Products **RTF2PDF** and **TextDynamic** also define this status items:
ENGINE_DLLUPTIME: seconds since loading the engine
ENGINE_UPTIME: seconds the engine object is alive
ENGINE_COUNT: total count of created engine objects since loading the engine
ENGINE_VERSION: the engine version, for example "3.50.5"
ENGINE_INFO: display dll-uptime, count, version and current date in one sentence.
SYSTEM_UPTIME: display the system uptime (using GetTickCount API)

When you need to create HTML text you can use a field with the name "HTMLCODE" and any custom code as "command" to insert custom HTML tags into the HTML text.

Example:

This code inserts page numbering objects:

```
IWPTextCursor TextCursor;
TextCursor = WpdllInt1.Memo.TextCursor;
TextCursor.InputFieldObject("PAGE", "", "1");
TextCursor.InputText("/");
TextCursor.InputFieldObject("NUMPAGES", "", "9");
```

Note: The following action names are available. They can be used with wpaProcess or in the menus defined inside the PCC file.

InsertTextFieldPAGE,
InsertTextFieldNEXTPAGE,
InsertTextFieldPRIORPAGE,
InsertTextFieldNUMPAGES,
InsertTextFieldSECTIONPAGES,
InsertTextFieldDATE,
InsertTextFieldTIME,

Note:
Also see function InputObject().

4.2.2.45 IWPTextCursor.InputFooter

Applies to
IWPTextCursor

Declaration
```
void InputFooter(int Range, string name, string Text)
```

Description
Create a footer with the specified "Range" selection. Please also see InputHeader and BlockAdd.

Parameters	

Range	wpraOnAllPages =0; // use on all pages
	wpraOnOddPages =1; // use on odd pages (1,3,5,...)
	wpraOnEvenPages=2; // use on even pages (2,4,6,...)
	wpraOnFirstPage=3; // print on first page only

	wpraOnLastPage=4; // print on last page only
	wpraNotOnFirstAndLastPages=5; // Not on first or last pages
	wpraNotOnLastPage=6; // on all but not on last page
	wpraNamed=7; // Use event OnGetSpecialText to select header or footer
	wpraIgnored=8; // Dont use this header or footer
	wpraNotOnFirstPage=9; // On all but not on first page

Name	A name for the footer, usually "" except for Range=wpraNamed
Text	The initialization text, if no text was specified the footer will be cleared and the cursor will be placed within. The text which is passed as parameter "Text" may be contain RTF or HTML formatting tags. When using HTML you can use the following proprietary closed HTML tags: **<pagenr/>:** Print the current page number **<pagecount/>:** Print the page count **<tab/>:** Insert a tabchar and create a tabstop: left, right, center, decimal=twips-value **<field/>:** create a merge field. Use name and command parameter.

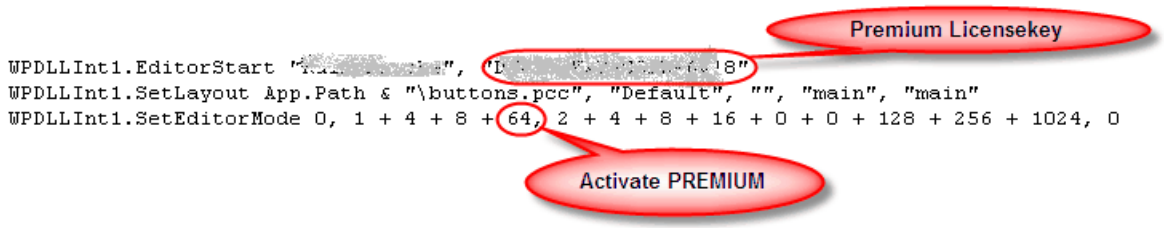
4.2.2.46 IWPTextCursor.InputFootnote

Applies to
IWPTextCursor

Declaration
```
int InputFootnote( int Mode, string Text)
```

Description
Creates a footnote - optionally places cursor in new footnote so you can use "Input" methods to add text.
Requires "premium" license.

Unless the footnote support (with premium license or demo) was activated with SetEditorMode, no footnote will be created!



Parameters

Mode (Bitfied)
1: PlaceCursorInFootnote
2 : CreateNumberInFootnote
4: NumberInFootnoteIsSuperScript

Text
Initialization text (may not contain HTML or RTF tags).

Returns

0 or the Obj ID of the created object.

Also see: InputTextBox

4.2.2.47 IWPTextCursor.InputHeader

Applies to
IWPTextCursor

Declaration
```
void InputHeader(int Range, string Name, string Text)
```

Description
Create a header with the specified "Range" selection.

Parameters Range, Name, Text have same meaning as for function InputFooter().

Please also see Memo.BlockAdd and Memo.BlockFind.

InputHeader Example

```
WPDLLInt1.TextCursor.InputHeader(
 0,   // OnAllPages
 "",  // no name
 // The text with HTML tags
 "Custom <b>Header</b>"+
 // Insert tab char and set right tab at 9000 twips
 "<tab right=9000/>"+
 // Insert page number text object
 "page <pagenr/>"+
 // Insert page count text object
 "/<pagecount/>");
```

Tip: To insert page numbers in code use IWPTextCursor.InputFieldObject

To delete a block (such a header or footer text) call the "Delete" method published by the interface `IWPDataBlock` which is returned by Memo.BlockFind.

4.2.2.48 IWPTextCursor.InputHTML

Applies to
IWPTextCursor

Declaration
```
procedure InputHTML(const Text: WideString);
```

Description
Inserts HTML formatted text at cursor position.

4.2.2.49 IWPTextCursor.InputHyperlink

Applies to
IWPTextCursor

Declaration
```
function InputHyperlink(const URL: WideString; const Text: WideString;
CursorWithin: WordBool): Integer;
```

Description
Creates a new hyperlink. You can specify the URL, the linked text and optionally place the

cursor witin the created link object pair.

Returns
0 if not successful. Otherwise the interface CurrObj can be used to change link properties.

Category
Hyperlinks and Bookmarks

4.2.2.50 IWPTextCursor.InputImage

Applies to
IWPTextCursor

Declaration
```
function InputImage(const filename: WideString; Mode: Integer): Integer;
```

Description
Inserts an image file at cursor position.

See method _SetObjType to convert an embedded image into a special image type which can also store attached data when exported to PDF. This can be useful to add the original source to a PDF file.

Parameters	
filename	The image file, may be BMP, WMF, EMF, JPG or PNG format.
Mode	If bit 1 is set the image will be linked, not embedded. If bit 2 is set the image will be positioned relatively to a paragraph (movable image). If bit 3 is set the image will be positioned relatively to a page (movable image). If bit 4 is set text will not wrapp around image (if a movable image). When bit 5 was set and the image was not found an error text object will be inserted. The error object will display <*filename*?>.

Returns
0 if not successful. Otherwise the interface CurrObj can be used to change image properties.

Tip

To create a text box use IWPTextCursor.InputTextbox. (Requires premium license)

Category
Image Support

4.2.2.51 IWPTextCursor.InputPicture

Applies to
IWPTextCursor

Declaration
```
int InputPicture(IUnknown Picture, int Width, int Height);
```

Description
Insert a picture. If you use the OCX you can use any IPicture reference.

Please also see InputPictureStream which is useful to insert dynamically created images.

When using .NET convert an "Image" using the Image2Picture class: TextCursor.InputPicture (new WPDynamic.Image2Picture(pictureBox1.Image),0,0) to get a reference usable for the "Picture" parameter.

Parameters
Picture : IPicture reference, or, with .NET, instance of WPDynamic.Image2Picture.
Width: 0 or the desired width in twips (= 1/1440 inch)
Height : 0 or the desired height in twips

Tip

To create a text box use IWPTextCursor.InputTextbox. (Requires premium license)

Returns
0 if not successful. Otherwise the interface CurrObj can be used to change imaget properties.

Category
Image Support

4.2.2.52 IWPTextCursor.InputPictureStream

Applies to
IWPTextCursor

Declaration
```
int InputPictureStream(IUnknown Stream, string FileExt, int Width, int Height);
```

Description
Insert a picture data specified as Stream.

Parameters:
Stream: If you use the OCX you can use any IStream reference.
When using .NET convert an "Stream" using the Stream2WPStream class: TextCursor.InputPictureStream(new WPDynamic.Stream2WPStream(stream1),".PNG",0,0);
FileExt: The file extension - it is used to use the correct loading algorithm. Values can be "BMP", "EMF", "WMF", "JPG" and "PNG"
Width: 0 or the desired width in twips (= 1/1440 twips)
Height: 0 or the desired height in twips

Returns
0 if not successful. Otherwise the interface CurrObj can be used to change image properties.

Example 1:

Create a file stream from an image file and insert it.

```
IWPTextCursor TextCursor = WPDLLInt1.Memo.TextCursor;
FileStream stream = new FileStream(
      "C:\\WPCubedLogo.jpg",FileMode.Open);
TextCursor.InputPictureStream(
      new WPDynamic.Stream2WPStream(stream),"jpg",
      0,0);
stream.Close();
```

Width and Height are always measured in twips (= 1/1440 inch). If 0, the content width and height will be used.

Example 2:

Create an image "on the fly" and insert it.

```
Bitmap aBitmap = new Bitmap(320, 200);
Graphics aGraphic = Graphics.FromImage(aBitmap);
System.IO.MemoryStream aStream = new System.IO.MemoryStream();
IWPMemo Memo = wpdllInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;

aGraphic.Clear(Color.AntiqueWhite);
SolidBrush RedBrush = new SolidBrush(Color.Red);
aGraphic.FillEllipse(RedBrush, 10, 10, 300, 180);

aBitmap.Save(aStream,System.Drawing.Imaging.ImageFormat.Bmp);
TextCursor.InputPictureStream(new WPDynamic.Stream2WPStream(aStream), "BMP", 0,
0);

// clean up
aStream.Dispose();
aGraphic.Dispose();
aBitmap.Dispose();
// ReleaseInt was added to TextDynamic V1.31
wpdllInt1.ReleaseInt(TextCursor);
wpdllInt1.ReleaseInt(Memo);
```

4.2.2.53 IWPTextCursor.InputObject

Applies to
IWPTextCursor

Declaration
`int InputObject(TextObjTypes ObjType, string Name, string Command, int Mode)`

Description
Creates different object kinds at cursor position. It can be used to create a single object and an object pair.

Note: If you need to create a "text object" you can also use InputFieldObject, for mail merge fields use InputField.

Parameters	
ObjType	**TextObjTypes:** 0=(default, will create "text object") 1=merge field, 2=hyperlink, 3=bookmark, 7=text object, 8=page reference, 11=footnote, 12=image, text box or other classes (see name) 13=horizontal lines (use IWPTextObj.SetProp(19, color) to change the color)
Name	The name of the new object. This can be the name of a page reference, bookmark or mail-merge field. "Text objects" can use the following predefined names: 'PAGE', 'NEXTPAGE', 'PRIORPAGE', 'NUMPAGES', 'DATE', 'TIME' and

	'SECTIONPAGES'. You can use other names but then have to use the event OnTextObjectGetText to provide the text which should be actually displayed. If You specify ObjType=12 You can specify a custom obj class name here. Please ask WPCubed for availability and customization of special objects.
Command	The command parameter of the new object. Hyperlinks use this parameter as url.
Mode	If bit 1 is set an object pair will be created. The cursor will be placed within both objects. Bit 2 activates the "editable" mode used for form fields. This must be combined with ObjType=1

Returns
0 if not successful. Otherwise the interface CurrObj can be used to change object properties.

Enum TextObjTypes:

```
Public Enum TextObjTypes
{
   wpobjCustom, // undefined
    // This objects are usually used pairwise
    wpobjMergeField, // Name=fieldname
    wpobjHyperlink, // Name=Title, Command = url
    wpobjBookmark, // Name=bookmark name
    wpobjTextProtection, // reserved
    wpobjSPANStyle, // Special texts styles
    wpobjCode, // reserved
    // This objects are usually used singular
    wpobjTextObject, // A Text Object field (one char Text, such As PAGE), Command=Mask)
    wpobjReference, // A Text Object field (Name=Bookmark, Command=Default Text)
    wpobjPageSize, // reserved
    wpobjPageProps, // reserved
    wpobjFootnote, // Only used with premium license. Name = name of text layer (RTFDataB
    wpobjImage, // an image
    wpobjHorizontalLine // a line (CParam=width, IParam = margin offset)
}
```

Example:

This code inserts page numbering objects:

```
IWPTextCursor TextCursor;
TextCursor = WpdllInt1.Memo.TextCursor;
TextCursor.InputObject(TextObjTypes.wpobjTextObject, "PAGE", "", 0);
TextCursor.InputText("/");
TextCursor.InputObject(TextObjTypes.wpobjTextObject, "NUMPAGES", "", 0);
```

This code inserts a blue thick line

```
IWPMemo Memo = wpdllInt1.Memo;
IWPTextCursor Cursor = Memo.TextCursor;
Cursor.InputObject(13, "", "", 0);
IWPTextObj obj = Memo.CurrObj;
if (obj != null)
{
    obj.SetProp(34, "blue"); // Set color value
```

```
            obj.SetProp(32, "-760"); // margin offset (outside of margins)
            obj.Height = 70;
        }
```

4.2.2.54 IWPTextCursor.InputCode

Applies to
IWPTextCursor

Declaration

int InputCode(int Mode: Integer; string Par1: WideString; string Par2: WideString);

Description

This Method creates a text object pair of type "wpobjCode". This type is reserved for XML editing.

Par1 will be assigned to the name, Par2 will be assigned to the "Source" property of the object.

The following mode bits can be set

1 : WrapSelectedText

2 : DeleteSelectedText

4: PlaceCursorAfterStart

8: DropMarkersOutside

16: DropMarkersInside

4.2.2.55 IWPTextCursor.InputPagebreak

Applies to
IWPTextCursor

Declaration
```
procedure InputPagebreak;
```

Description
Creates a page break. If the current position is not at the start of a paragraph it will first create a paragraph break at the current position and insert the page break before the new paragraph.

InputPagebreak Example

Merge RTF files and create table-of-contents
The Text uses _TocXXXX Bookmarks around headline text, in the RTF code this reads as
{\bkmkstart _Toc133939675}Section Header{*\bkmkend _Toc133939675}*

```
wpdllInt1.Memo.TextCursor.GotoStart();
wpdllInt1.Memo.SetBProp(WPDynamic.BPropSel.wpWriteObjectMode,5,1);
wpdllInt1.Memo.wpaProcess("InsertToc","");
wpdllInt1.Memo.TextCursor.InputPagebreak();
wpdllInt1.Memo.LoadFromFile(@"C:\Title.rtf",true,"RTF");
wpdllInt1.Memo.LoadFromFile(@"C:\Body.rtf",true,"RTF");
wpdllInt1.Memo.TextCursor.InputPagebreak();  wpdllInt1.Memo.LoadFromFile(@"C:\Title2.rtf"
wpdllInt1.Memo.LoadFromFile(@"C:\Body2.rtf",true,"RTF");
```

4.2.2.56 IWPTextCursor.InputParagraph

Applies to
IWPTextCursor

Declaration
```
procedure InputParagraph(Mode: Integer; const StyleName: WideString);
```

Description
Creates a new paragraph and places the cursor on it.

By default the paragraph is created after the current.

Parameters	
Mode	Parameter Mode is a bit field - please combine the required values with "+". 1 : Start a new page before the new paragraph. 2 : If currently in table, append the new paragraph after the table. (Alternative: IWPTextCursor.ExitTable) 4 : Append the paragraph **at the very end of the text.**
StyleName	This is a string: Paragraph style used by new paragraph

If bit 4 or 8 was used, the new paragraph can be modified by CurrPar. The text will not be automatically reformat so please call ReformatAll when You are done.

Also see CurrPar.ParStrCommand - it can be used to modify the created paragraph, for example to assign a name.

Category
Paragraphstyle Support

4.2.2.57 IWPTextCursor.InputRowEnd

Applies to
IWPTextCursor

Declaration

```
procedure InputRowEnd;
```

Description
Closes a row which was started with InputRowStart.

Category
Table Support

4.2.2.58 IWPTextCursor.InputRowStart

Applies to
IWPTextCursor

Declaration

```
procedure InputRowStart(Mode: Integer);
```

Description
Start a new row. This is only possible after InputTable or after InputRowEnd.

Parameters	
Mode	Bit 1: Add borders to all cells.

This VB.NET example creates a table with merged cells and alternated use of paragraph styles:

¶

This is the header text		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

¶

```
Dim Memo As IWPMemo
Dim TextCursor As IWPTextCursor
Dim Cell As IWPParInterface

Memo = WpdllInt1.Memo
TextCursor = Memo.TextCursor
Cell = Memo.CurrPar
```

```
'Initialize Styles
```

```
Dim headsty As String = "TableHeaderStyle"
Memo.SelectStyle(headsty)
Memo.CurrStyle.ParColor = WpdllInt1.ColorToRGB(Color.Black)
Memo.CurrStyle.ParShading = 30 ' 30% shading
'Style for odd rows (is empty)
Memo.SelectStyle("DataOdd")
```

```
'Style for even rows (shaded)
Memo.SelectStyle("DataEven")
Memo.CurrStyle.ParColor = WpdllInt1.ColorToRGB(Color.Blue)
Memo.CurrStyle.ParShading = 20 ' 20% blue


'Use the default attributes
Memo.CurrAttr.Clear()
```

'Create a table

```
TextCursor.InputTable(0, "")
TextCursor.InputRowStart(1) ' with border
' Now 3 cells merged as  two one
TextCursor.InputCell("This is the header text", headsty)
Cell.Alignment = 1 'center
TextCursor.InputCell("", headsty)
Cell.IsColMerge = True
TextCursor.InputCell("", headsty)
Cell.IsColMerge = True
TextCursor.InputRowEnd()

' Now some rows below, 3 columns each
Dim i As Integer = 1
Dim sty As String
While i <= 10
    If (i And 1) = 0 Then
        sty = "DataEven"
    Else
        sty = "DataOdd"
    End If
    TextCursor.InputRowStart(1)
    TextCursor.InputCell(i, sty)
    Cell.ParASet(WPAT.COLWIDTH_PC, 10 * 100) ' 10%
    Cell.ParAAddBits(WPAT.CharStyleON, 1) 'Bold
    Cell.ParAAddBits(WPAT.CharStyleMask, 1) 'Bold
    TextCursor.InputCell("", sty)
    Cell.ParASet(WPAT.COLWIDTH_PC, 45 * 100) ' 45%
    TextCursor.InputCell("", sty)
    Cell.ParASet(WPAT.COLWIDTH_PC, 45 * 100) ' 45%
    TextCursor.InputRowEnd()
    i = i + 1
End While
```

' Exit the table

```
TextCursor.InputParagraph(2, "")

'Display the text
Memo.ReformatAll(False, True)
```

Category
Table Support

4.2.2.59 IWPTextCursor.InputSection

Applies to
IWPTextCursor

Declaration
```
IWPPageSize InputSection(int Mode);
```

Description
Creates a new section in the text or locates it for modification.

It returns a reference to the section reference. You need to call ReleaseInt for the returned interface!

Parameter Mode:

a) The value -1 will locate the section which is valid at the current position (may be started in any paragraph before).
In this case the result value can be null! You can use IWPPageSize.GetProp(0) to red the section id of this section. This id can be then used in Memo.BlockAdd to create a header or footer for a specific section.

b) Bitfield:
If bit 2 is set a new paragraph is added after a table.
Bit 1 sets a page break.

Example:
```
// a) Create
IWPPageSize sect;
sect = wpdllInt1.TextCursor.InputSection(1);
sect.Landscape = true;
wpdllInt1.ReleaseInt(sect);

// b) Modify
IWPPageSize sect;
sect = wpdllInt1.TextCursor.InputSection(-1);
if (sect!=null)
{
   sect.Landscape = true;
   wpdllInt1.ReleaseInt(sect);
}
```

Note: the select flag (use SetProp/GetProp) to set and read it will be automatically set when a property is changed.

Returns
Reference to IWPPageSize interface to modify the page size used by this section.

4.2.2.60 IWPTextCursor.InputString

Applies to
IWPTextCursor

Declaration
```
procedure InputString(const Text: WideString; CharAttrIndex: Integer);
```

Description
Inserts a string at cursor position

Parameters	
Text	The text to be inserted

CharAttrIndex	-1 to insert the text neutral, without any character attributes 0 to use current writing mode (=InputText) >0 to use the specified CharacterAttr index.

Tip:
To read the character at the current position use
`Memo.CurrPar.GetChar(Memo.TextCursor.CPPosInPar)`

Example:
```
// We need this interfaces for text creation
IWPMemo Memo = WPDLLInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;
IWPAttrInterface AttrHelper = WPDLLInt1.AttrHelper;

// Calculate a character style index value
AttrHelper.Clear();
AttrHelper.SetFontface("Times New Roman");
AttrHelper.SetFontSize(11);
AttrHelper.IncludeStyles(2); // Italic
int mycharattr = AttrHelper.CharAttrIndex;

// and insert some text
TextCursor.InputString("Hello World", mycharattr);
```

4.2.2.61 IWPTextCursor.InputTable

Creates and fills a table not using a callback

Applies to
IWPTextCursor

Declaration
```
void InputTable(int Width, string Name);
```

Description
Creates a table object. Create each row with InputRowStart followed by some InputCell and finished by InputRowEnd. Use InputParagraph to create text after the table. You can use MoveToTable to later locate the table.

Note: Please make sure to create an empty paragraph at the end of a text - otherwise some RTF readers have problems to load it. To do so use InputParagraph when the table is complete.

This C# example creates a table with 10 rows and 3 columns:

```
IWPMemo Memo;
Memo = WPDLLInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;
IWPAttrInterface CurrParAttr = Memo.CurrParAttr;
TextCursor.InputTable(0, "data");
For (int r = 0; r < 10; r++)
{
       TextCursor.InputRowStart(1);
       For(int c = 0; c < 3; c++)
       {
        TextCursor.InputCell("some text", "");
        // This cell should use bold Text
        If(c==0)CurrParAttr.IncludeStyles(1);
```

```
        }
        TextCursor.InputRowEnd();
}
TextCursor.InputParagraph(0,"");
Memo.Reformat();
```

This code creates a table by appending one row and one cell after each other. After the creation of each cell this cell can be modified using the CurrParAttr and CurrPar interfaces.

The table is created named as "data". So you can use TextCursor.**MoveToTable**("data") to move the cursor to the first cell in this table anytime later.

You can use CurrPar and CurrParAttr to modify the tables, rows and cells right after the have been created.

This C# example creates a table with 5 rows and 2 cells, width = 10 and 90%:

```
IWPMemo memo = wpdllInt1.Memo;
IWPAttrInterface atr = wpdllInt1.AttrHelper;
IWPParInterface par = memo.CurrPar;
IWPAttrInterface parattr = memo.CurrParAttr;
IWPTextCursor cursor = memo.TextCursor;
// Start a table
 cursor.InputTable(0,"");
// use 50 % of the page width
 par.ParASet((int)WPAT.BoxWidth_PC, 50*100);
// Now create 5 rows
 For (int r = 1; r <= 5; r++)
 {
    cursor.InputRowStart(0);
    // With 2 cells Each, 10 And 90 % width
     cursor.InputCell(r.ToString(),"");
     par.ParASet((int)WPAT.COLWIDTH_PC, 10*100);
     par.ParColor = wpdllInt1.ColorToRGB(Color.Gray);
     parattr.IncludeStyles(1); // bold Text
    // The second cell uses different Text attributes
     cursor.InputCell("","");
     par.ParASet((int)WPAT.COLWIDTH_PC, 90*100);
    // Append normal Text
     atr.Clear();
     par.AppendText("Normal ", atr.CharAttrIndex);
     atr.IncludeStyles(1); // bold Text
     par.AppendText("and bold", atr.CharAttrIndex);
    // This row is finished
     cursor.InputRowEnd();
 }
// Format And display
 memo.ReformatAll(False, True);
```

Category
Table Support

4.2.2.62 IWPTextCursor.InputTabstop

Applies to
IWPTextCursor

Declaration

```
procedure InputTabstop(ClearAllTabs: WordBool; Value: Integer; Kind: Integer;
```

```
FillMode: Integer);
```

Description

This method creates a tab-stop and also inserts the tab character. It can optionally clear the existing tabstops. If you only want to define a tab-stop but do not need to insert the tab character use the method CurrPar.TabAdd.

Parameters	
ClearAllTabs	True to clear tab list for current paragraph.
Value	Value of this tabstop (in twips=1/1440 inch)
Kind	The tabstop kind: 0 : Left tab 1 : Right tab 2 : Center tab 3 : Decimal tab
Fill	The fill mode used for this tabstop 0: No Filling 1: Dots ..... 2: Dots in middle of line 3: Hyphens ------ 4: Underline _____ 5: Thick Hyphen **---------** 6: Equal Signs ======= 7: Arrow

This VB.NET example code inserts page numbering objects aligned at the right margin of the page. You can use it to add page numbers in a header or footer texts.

```
      Dim Memo As IWPEditor
      Memo = WpdllInt1.Memo
      Dim TextCursor As IWPTextCursor
      TextCursor = Memo.TextCursor

      TextCursor.InputText(" Page ")
      TextCursor.InputTabstop(False, _
            Memo.PageSize.PageWidth - Memo.PageSize.LeftMargin - Memo.PageSize.
RightMargin, _
            1, 1)
      TextCursor.InputObject(TextObjTypes.wpobjTextObject, "PAGE", "", 0)
      TextCursor.InputText(" of ")
      TextCursor.InputObject(TextObjTypes.wpobjTextObject, "NUMPAGES", "", 0)
```

4.2.2.63 IWPTextCursor.InputTextbox

Applies to
IWPTextCursor

Declaration

```
int InputTextbox(int Width, string Text);
```

Description
Creates a text box. If no text was specified the cursor will be placed within the text object layer. You can use InputText to add more text. This feature requires the "premium" license, it is always available in RTF2PDF.

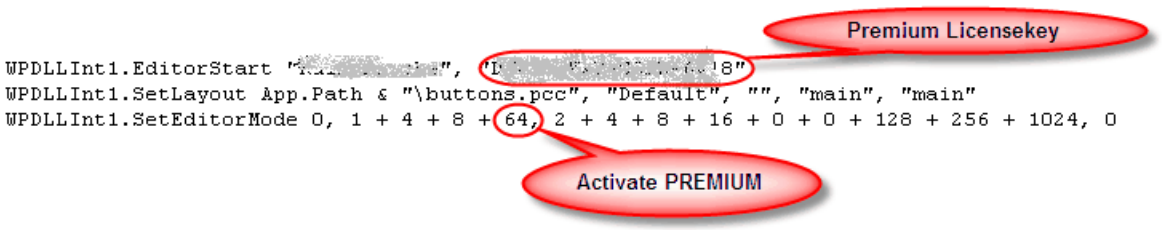
After the insertion of the text box you can set the relative location using

Memo.CurrObj.RelX = twips value;
Memo.CurrObj.RelY = twips value;

Using CurrObj.Frame You can select the type of frame (border) which is drawn for the box.

Please note that after GotoBody the interface CurrObj cannot be used anymore.

Unless the text box support (with premium license or demo) was activated with SetEditorMode, no textbox will be created!



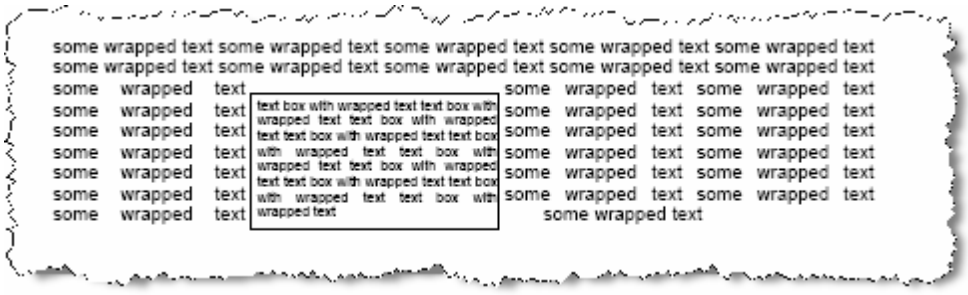
Parameters:
Width : The Width of the box in twips. The height is always determined by the contained text.
Text : The initialization text - may contain HTML or RTF tags. To create an empty box use "CLEAR". If no text was specified (except "CLEAR") the cursor will be places inside the new text box. You can fill in text and then call GotoBody when you are done.

Returns
0 if not successful. Otherwise the interface CurrObj can be used to change object properties.

Tips:
Using CurrObj.property Wrap you can change the way the text wraps around the box.

RelX and RelY are the offset coordinates (in twips) to the anchor point. By default is the anchor point the origin of the current paragraph. If you want to make the current page the origin use PositionMode = 2

Example:




```
TextCursor.InputTextbox(2880, "");
// Save writing mode, then enter some text
int save_ch = Memo.CurrAttr.CharAttrIndex;
Memo.CurrAttr.SetFontSize(8);
for (int i = 0; i < 10; i++)
TextCursor.InputText("text box with wrapped text ");
Memo.CurrPar.Alignment = 3;  // justified text
Memo.CurrObj.RelX = 3500;
Memo.CurrObj.RelY = 720;
Memo.CurrObj.Wrap = 4;
```

```
Memo.CurrObj.Frame = 1;
TextCursor.GotoBody();
Memo.CurrAttr.CharAttrIndex = save_ch;
Memo.CurrPar.Alignment = 3;
for (int i = 0; i < 30; i++)
TextCursor.InputText("some wrapped text ");
```

VB Example:

```
Dim Memo As IWPMemo
Dim Cursor As IWPTextCursor
Set Memo = WPDLLInt1.Memo
Set Cursor = Memo.TextCursor

'find out how to set to set the relx to the actual cursor position.
Dim x As Long
Dim y As Long
Dim page As Long
' this would check the mouse coordinates
' Memo.GetMouseXY page, x, y

' this checks the text cursor position
Memo.GetXY 3, x, y


Cursor.InputTextbox 2000, "TextBox"
ActiveForm.WPDLLInt1.Memo.CurrObj.RelX = x
ActiveForm.WPDLLInt1.Memo.CurrObj.RelY = y

' relatively to page
ActiveForm.WPDLLInt1.Memo.CurrObj.PositionMode = 2
```

4.2.2.64 IWPTextCursor.InputText

Applies to
IWPTextCursor

Declaration
```
procedure InputText(const Text: WideString);
```

Description
Writes some text suing current writing attributes. You can use InputString if you want to specify a character attribute index.

4.2.2.65 IWPTextCursor.InsertRow

Applies to
IWPTextCursor

Declaration
```
function InsertRow(Before: WordBool): WordBool;
```

Description
Insert a row after the current row. With parameter "Before"=true the row will be inserted before the current.

The text in the row is not duplicated unless the mode DuplicateWithText has been activated using Memo.SetBProp(18,1);

Tip: A table row can also be duplicated using the CurrPar interface. Here we first let the CurrPar interface modify the current row and then call Duplicate():

```
Cursor.CPTableRowNr = 1; // goto row 1
Memo.CurrPar.SetPtr( Cursor.CPRowPtr );
while(rowcount-->0) Memo.CurrPar.Duplicate();
```

To delete the current row use the Action:

```
TD.wpaProcess("DelRow")
```

4.2.2.66 IWPTextCursor.MarkerCollect

Applies to
IWPTextCursor

Declaration
```
procedure MarkerCollect(ID: Integer);
```

Description
Remove the marker with the given ID.

Category
Position Markers

4.2.2.67 IWPTextCursor.MarkerCollectAll

Applies to
IWPTextCursor

Declaration
```
procedure MarkerCollectAll;
```

Description
Remove all markers.

Category
Position Markers

4.2.2.68 IWPTextCursor.MarkerCPPosition

Applies to
IWPTextCursor

Declaration
```
function MarkerCPPosition(ID: Integer): Integer;
```

Description

Calculate the character position which matches the marker with the given ID.

Category
Position Markers

4.2.2.69 IWPTextCursor.MarkerDrop

Applies to
IWPTextCursor

Declaration
```
int MarkerDrop(int Mode);
```

Description
Insert a marker and return the ID.

Mode can be
0 : drop a maker at the cursor position
1 : drop the marker at the start of the selection
2 : drop the marker at the end of the selection

Category
Position Markers

4.2.2.70 IWPTextCursor.MarkerGoto

Applies to
IWPTextCursor

Declaration
```
void MarkerGoto(bool Collect; int ID);
```

Description
Move the cursor to a marker, optionally collect it.

Category
Position Markers

4.2.2.71 IWPTextCursor.MarkerSelect

Applies to
IWPTextCursor

Declaration
```
procedure MarkerSelect(StartID: Integer; EndID: Integer);
```

Description
Select the text between marker with ID#1 and ID#2.

Category
Position Markers

4.2.2.72 IWPTextCursor.MovePosition

Applies to
IWPTextCursor

Declaration
```
void MovePosition(int MoveMode, bool SelectText);
```

Description
Moves the cursor, optionally select text.

Parameters	
MoveMode	0 = LineStart 1 = Line End 2 = Page Up 3 = Page Down 4 = Start of Text (Home) 5 = End of Text 6 = Cursor left 7 = Cursor right 8 = Cursor up 9 = Cursor down 10= Word Left 11= Word Right 12= End of Selection 13= Start of Selection 14= Start of current merge field 15= End of current merge field
SelectText	If parameter SelectText = true the text will be selected from current to new position.

4.2.2.73 IWPTextCursor.MoveToBookmark

Applies to
IWPTextCursor

Declaration
```
function MoveToBookmark(const Name: WideString): WordBool;
```

Description
Moves the cursor to the specified bookmark. Returns true if bookmark was found, false if it was not found.

Category
Hyperlinks and Bookmarks

4.2.2.74 IWPTextCursor.MoveToField

Applies to
IWPTextCursor

Declaration
bool MoveToField(string **Name**, int **Mode**, int **Option**);

This is a versatile methods to jump to and between mail merge fields:

Parameters:

Mode = 0:
Locate the first field in the text, or the first field with the provided name if parameter "name" was not empty.

Mode = 1:
Locate the next field in the text, or the next field with the provided name if parameter "name" was not empty.

Mode = 2:
Locate previous field

Mode = 3
Locate the last field

Option - Bitfield:
If bit 1 was set in parameter "option" the cursor will be positioned inside the field at the start: <|....>

If bit 2 was set in parameter "option" the cursor will be positioned after the end of the field: <....>|

If bit 1 and 2 are set the cursor will be at the end INSIDE the field: <....|>

If Bit 4 and 2 was set, the field will be selected. <....|>

Examples:
Select the field and the marker:

```
wpdllInt1.TextCursor.MoveToField("FELD", 0, 6);
```

Select only the text of the field:

```
wpdllInt1.TextCursor.MoveToField("FELD", 0, 7);
```

This example will move to next field (from current position).

```
wpdllInt1.Memo.TextCursor.MoveToField("",1,1));
```

Also see: Memo.TextCommandStr ID 43-46: Delete, Read and Write a field and also Load and Save Categor

Please also see Category Mailmerge and the "mail merge API introduction".

4.2.2.75 IWPTextCursor.MoveToObject

Applies to
IWPTextCursor

Declaration
bool MoveToObject(int ObjType, string Name);

This method can be used to find a certain object with the given type and the given name.

4.2.2.76 IWPTextCursor.MoveToTable

Applies to
IWPTextCursor

Declaration

```
bool MoveToTable(string Name);
```

Description

Moves to a table with a given name. The methods AddTable and InputTable allow it to give the table a name. If no table was found the result value will be false.

```
if (TextCursor.MoveToTable("MYTABLE"))
{
   int val, sum = 0,i = 1;
   Random rnd = new Random();
   while(true)
   {
      TextCursor.CPTableColNr = 0; // First Cell
      TextCursor.InputText(i.ToString());
      TextCursor.CPMoveNextCell();
      val = rnd.Next(1000);
      TextCursor.InputText(val.ToString());
      TextCursor.CPMoveNextCell();
      sum+=val;
      TextCursor.InputText(sum.ToString());
      if (!TextCursor.CPMoveNextRow(false)) break;
   }
}
```

You may pass an empty string to locate the first table in the document. "{NESTEDTABLE}" will locate the first nested table in the current table cell. "{NEXTTABLE}" will locate the next table within the current nesting level. In case a table was found the cursor will be always moved into the first cell of the table.

Tip: You can use the methods ASetCellProp and ASetCellStyle to quickly modify the properties, text and styles of certain cells in an existing table.

To move to other named paragraphs use Memo.TextCommandStr(42, 0, name).

4.2.2.77 IWPTextCursor.Redo

Applies to
IWPTextCursor

Declaration
```
function Redo: WordBool;
```

Description
Undoes the last UNDO action.

Category
Standard Editing Commands

4.2.2.78 IWPTextCursor.ReplaceText

Applies to
IWPTextCursor

Declaration
```
function ReplaceText(const Text: WideString; const NewText: WideString;
FromStart: WordBool; CaseSensitive: WordBool; WholeWords: WordBool; ReplaceAll:
WordBool): Integer;
```

Description

Replaces Text

Parameters	
Text	The text to be replaced
NewText	The replacement text
FromStart	Start at the beginning of the text
CaseSensitive	Work case sensitive
WholeWords	Replace whole words only
ReplaceAll	Replace all

Returns
The count of replacements

4.2.2.79 IWPTextCursor.ReportConvertTable

Applies to
IWPTextCursor

Declaration
```
procedure ReportConvertTable(const GroupName: WideString; Options: Integer);
```

Description
Convert the current table into a reporting group.

Parameters	
GroupName	The name for the group
Options	Reserved

Category
Reporting

4.2.2.80 IWPTextCursor.ReportConvertText

Applies to
IWPTextCursor

Declaration
```
procedure ReportConvertText(const GroupName: WideString; Options: Integer);
```

Description
Convert the current document into a reporting group.

Parameters	
GroupName	The name for the group
Options	Reserved

Category
Reporting

4.2.2.81 IWPTextCursor.ReportInputBand

Applies to
IWPTextCursor

Declaration
```
procedure ReportInputBand(Typ: Integer; const Name: WideString; Options:
Integer);
```

Description
__RESERVED__

Category
Reporting

4.2.2.82 IWPTextCursor.ReportInputGroup

Applies to
IWPTextCursor

Declaration
```
procedure ReportInputGroup(const Name: WideString; Options: Integer; Were:
Integer);
```

Description
__RESERVED__

Category
Reporting

4.2.2.83 IWPTextCursor.ScrollToCP

Applies to
IWPTextCursor

Declaration
```
void ScrollToCP();
```

Description

Scrolls to the current paragraph.

4.2.2.84 IWPTextCursor.SelectAll

Applies to
IWPTextCursor

Declaration
```
procedure SelectAll;
```

Description
Select the complete text

Category
Standard Editing Commands

4.2.2.85 IWPTextCursor.SelectLine

Applies to
IWPTextCursor

Declaration

```
procedure SelectLine;
```

Description
Select the current line

Category
Standard Editing Commands

4.2.2.86 IWPTextCursor.SelectParagraph

Applies to
IWPTextCursor

Declaration
```
procedure SelectParagraph;
```

Description
Select the current paragraph

Category
Standard Editing Commands

4.2.2.87 IWPTextCursor.SelectTable

Applies to
IWPTextCursor

Declaration
```
procedure SelectTable;
```

Description
Select the current table

Category
Standard Editing Commands
Table Support

4.2.2.88 IWPTextCursor.SelectTableColumn

Applies to
IWPTextCursor

Declaration
```
procedure SelectTableColumn;
```

Description
Select the current table column

Category
Table Support

4.2.2.89 IWPTextCursor.SelectTableRow

Applies to
IWPTextCursor

Declaration
```
procedure SelectTableRow;
```

Description
Select the current table row

Category
Table Support

4.2.2.90 IWPTextCursor.SelectText

Applies to
IWPTextCursor

Declaration
```
procedure SelectText(SelStart: Integer; SelEnd: Integer);
```

Description
Select text range.

Parameters	
SelStart	Select from here
SelEnd	to here (not including)

4.2.2.91 IWPTextCursor.SetColWidth

Applies to
IWPTextCursor

Declaration
```
function SetColWidth(Mode: Integer; Value: Integer; IsPercent: WordBool):
WordBool;
```

Description
Modify the width of the current column.

Parameters	
Mode	Currently only Mode=0 is supported
Value	Value as twips or Percent
IsPercent	TRUE to select Value as Precent The Percent Value has to be multiplied with 100 (see WPAT_ColWidth_PC)

4.2.2.92 IWPTextCursor.SetParName

Applies to
IWPTextCursor

Declaration
```
function SetParName(ParagraphPtr: Integer; const Name: WideString): WordBool;
```

Description
Set the name of the paragraph with the given ID.

ParagraphPtr must be a valid paragraph ID or one of the following constants:

1: Assigns the name of the current paragraph

2: Assigns the name to the parent table of this paragraph. Returns false if not in a table. The table name can be used in MoveToTable.

3: Assigns the name to the parent-parent table of this paragraph. Returns false if not in a nested table.

Note: CurrPar.ParStrCommand can also change table or paragraph names.

Also see: IWPTextCursor.GetParName

4: Set the WPAT_Par_Name of the parent table

5: Set the WPAT_Par_Name of the parent-parent table

6: Set the WPAT_Par_Command of the parent table

7: Set the WPAT_Par_Command of the parent-parent table

4.2.2.93 IWPTextCursor.SetRowHeight

Applies to
IWPTextCursor

Declaration
```
bool SetRowHeight(int Mode, int MinHeight, int MaxHeight);
```

Description
Set the height of the current row or all rows in the table.

Parameters

Mode	0: change only current row 1: change all rows in current table 2: change rows which have selected cells.
MinHeight	0 or the minimum height in twips
MaxHeight	0 or the maximum heigt in twips

Important
Please make sure that either MinHeight or MaxHeight is 0 since in RTF format only one of this values can be stored (tag: \trrh).

4.2.2.94 IWPTextCursor.SetTableLeftRight

Change width and margins.

Applies to
IWPTextCursor

Declaration
```
bool SetTableLeftRight(int LeftTW, int RightTW, int WidthTW, int WidthPC);
```

Description
Change the width and the margins of the current table.

Parameters

LeftTW	-1 or the new left offset in twips
RightTW	-1 or the new right offset in twips

WidthTW	0 or the new width in twips
WidthPC	0 or the new eidth in percent

Category
Table Support

4.2.2.95 IWPTextCursor.TableClear

Initialize a table.

Applies to
IWPTextCursor

Declaration
```
procedure TableClear(KeepHeader: Integer; KeepFooter: Integer; KeepBody:
Integer; KeepProtected: WordBool; KeepFields: WordBool; KeepColsLeft: Integer;
KeepColsRight: Integer);
```

Description
Versatile function to initialize the current table. It can delete not required table body rows, clear required body rows but protect header and footer rows.

Parameters	
KeepHeader	Count of header rows which should not be cleared.
KeepFooter	Count of footer rows which should not be cleared.
KeepBody	Count of body rows which should not be deleted.
KeepProtected	If true the cells which are protected will be not modified.
KeepFields	If true fields will not be deleted.
KeepColsLeft	Number of columns to the left which will not be modified.
KeepColsRight	Number of columns to the right which will not be modified.

Category
Table Support

4.2.2.96 IWPTextCursor.TableDelete

Applies to
IWPTextCursor

Declaration
```
function TableDelete: WordBool;
```

Description
Deletes the current table.

Category
Table Support

4.2.2.97 IWPTextCursor.Undo

Applies to
IWPTextCursor

Declaration
```
function Undo: WordBool;
```

Description
Undeos the last change to the text.

Category
Standard Editing Commands

4.2.2.98 IWPTextCursor.UndoClear

Applies to
IWPTextCursor

Declaration
```
procedure UndoClear;
```

Description
Clears the undo buffer.

Category
Standard Editing Commands

4.2.2.99 IWPTextCursor.WordCharAttr

Applies to
IWPTextCursor

Declaration
```
function WordCharAttr(const AWord: WideString; CaseSensitive: WordBool;
OnylActiveText: WordBool; CharAttrIndex: Integer): Integer;
```

Description
Method to assign a certain character attribute index to certain words in the text.

Parameters	
AWord	The word to search for
CaseSensitive	true to work case sensitive
OnlyActiveText	true to only work in current text block
CharAttrIndex	New character attribute index

Category
Character Attributes

4.2.2.100 IWPTextCursor.WordEnum

Applies to
IWPTextCursor

Declaration
```
function WordEnum(const AWord: WideString; CaseSensitive: WordBool;
OnlyActiveText: WordBool; ParamForEvent: LongWord): Integer;
```

Description
Triggers the event OnEnumParOrStyle for all occurances of the given word. (A "word" is text which is enclosed in space or punctation characters!)

C# Example to make all occurances of "td" (not case sensitive) bold and replace it with

"TextDynamic".

```
private void testbutton_Click(
 object sender, System.EventArgs e)
{
 WPDLLInt1.TextCursor.WordEnum( "td",false,true,0);
 WPDLLInt1.Memo.ReformatAll(true,true);
}

private void WPDLLInt1_OnEnumParOrStyle(
 object Sender,
 bool IsControlPar,
 int StartPos,
 int Count,
 WPDynamic.IWPParInterface ParText,
 WPDynamic.IWPAttrInterface ParAttr,
 int EventParam,
 ref bool Abort)
{
  string replacestr = "TextDynamic";
  for(int i=0;i<Count;i++)
  {
    // Update the style (add bold)
    ParText.CharAttr(StartPos+i).IncludeStyles(1);
  }
  // Replace text (replacestr may have different length)
  ParText.ReplaceText(StartPos,Count,replacestr);
}
```

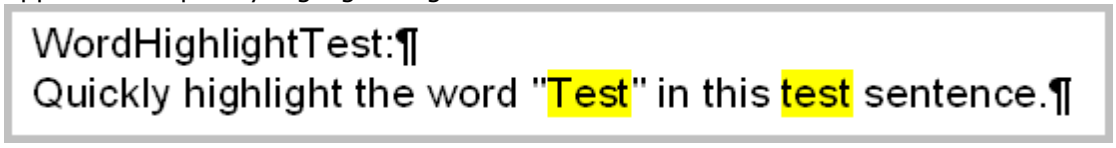
Parameters	
AWord	The word to search for
CaseSensitive	true to work case sensitive
OnlyActiveText	true to only work in current text block
ParamForEvent	Integer value passed through to event

4.2.2.101 IWPTextCursor.WordHighlight

Applies to
IWPTextCursor

Declaration
```
function WordHighlight(const AWord: WideString; CaseSensitive: WordBool;
OnlyActiveText: WordBool; RemoveHighlight: WordBool): Integer;
```

Description
Applies a temporary highlight flag to certain words




```
wpdllInt1.TextCursor.WordHighlight("test",
 false,false,!checkBox1.Checked);
```

Parameters	
AWord	The word to search for
CaseSensitive	true to work case sensitive

OnlyActiveText	true to only work in current text block
RemoveHighlight	false to set highlight, true to remove highlights

4.2.2.102 IWPTextCursor.SetColumns

Applies to
IWPTextCursor

Declaration
```
int SetColumns( int Command,  int Value);
```

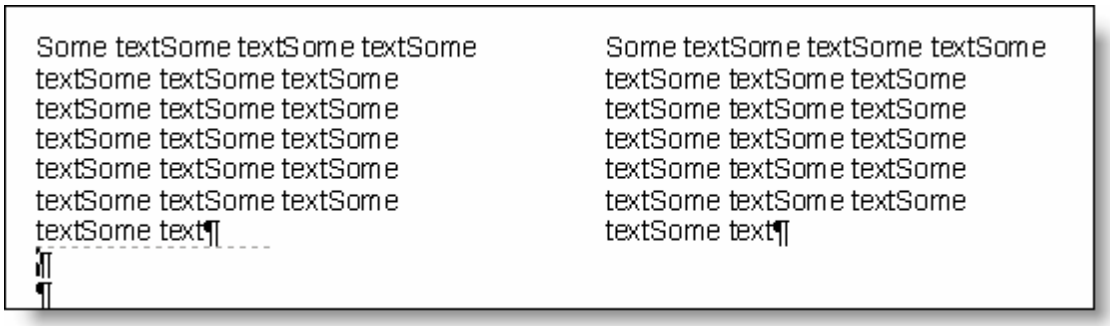
Description
Applies properties to control the text column feature (not table columns). You need the "Premium" license to use this feature.

The following command IDs are supported:

0 : set the number of columns, use 1 to switch columns off.
1 : set the horizontal distance of the columns (in twips)
2 : set the vertical height of the column area (in twips)
3 : toggles the "next column" flag
4 : sets the "next column" flag
5 : deletes the "next column" flag

Example - create 2 columns with text

```
IWPMemo Memo = wpdllInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;

TextCursor.SetColumns(0, 2);
for (int i = 0; i < 20; i++)
   TextCursor.InputText("Some text");
TextCursor.InputText("\r");
TextCursor.SetColumns(4, 0);
for (int i = 0; i < 20; i++)
   TextCursor.InputText("Some text");
TextCursor.InputText("\r");
TextCursor.SetColumns(0, 1);

Memo.Reformat();
```



4.2.2.103 IWPTextCursor.SelectCell

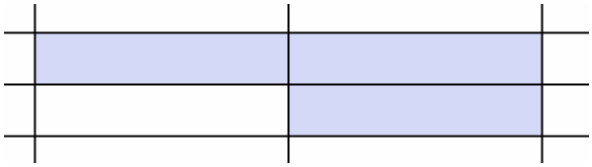
Applies to
IWPTextCursor

Declaration
```
void SelectCell( bool AddToSelection);
```

Description
Activate the selection mode for the current cell. It is possible to select several cells to apply an attribute using CuurSelAttr.

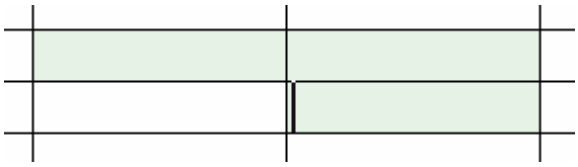
Part 1: Select cells

```
TextCursor.SelectCell(true);
TextCursor.CPMoveNextCell();
TextCursor.SelectCell(true);
TextCursor.CPMoveNextRow(false);
TextCursor.SelectCell(true);
```



Part 2: Set Shading

```
Memo.CurrSelAttr.AttrSET((int)WPAT.ShadingValue, 10);
Memo.CurrSelAttr.AttrSET((int)WPAT.FGColor, 2);
TextCursor.HideSelection();
```



4.2.2.104 IWPTextCursor.EnumOpenObj

Applies to
IWPTextCursor

Declaration
```
int EnumOpenObj( int ObjType, int EventParam);
```

Description
This method calls the event OnEnumTextObj for all open markers at the current position.

4.2.2.105 IWPTextCursor.ExitTable

Applies to
IWPTextCursor

Declaration
```
void ExitTable();
```

Description
Leaves the current table. The cursor is located in the next paragraph after the table. If necessary a new paragraph will be created.

```
TextCursor.ExitTable();
```

```
TextCursor.InputText("Text after the table");
```

4.2.2.106 IWPTextCursor.TableSplit

Applies to
IWPTextCursor

Declaration
```
bool TableSplit(bool BeforeRow);
```

Description
Splits the table after or before the current row.

Example:
```
TextCursor.TableSplit(false);
TextCursor.ExitTable();
TextCursor.InputText("Text between tables");
Memo.ReformatAll(false, true);
```

4.2.2.107 IWPTextCursor.TableSort

Applies to
IWPTextCursor

Declaration
```
bool TableSort(int HeaderRows, int FooterRow, int SelectColumn, int Method);
```

Description
This method sorts the rows in a table.

The given count of header and footer rows are not sorted. You can pass -1 to let header and footer rows be auto detected.

Header	
qweqw	
aaaa	
zzzzz	
a111	
Footer	

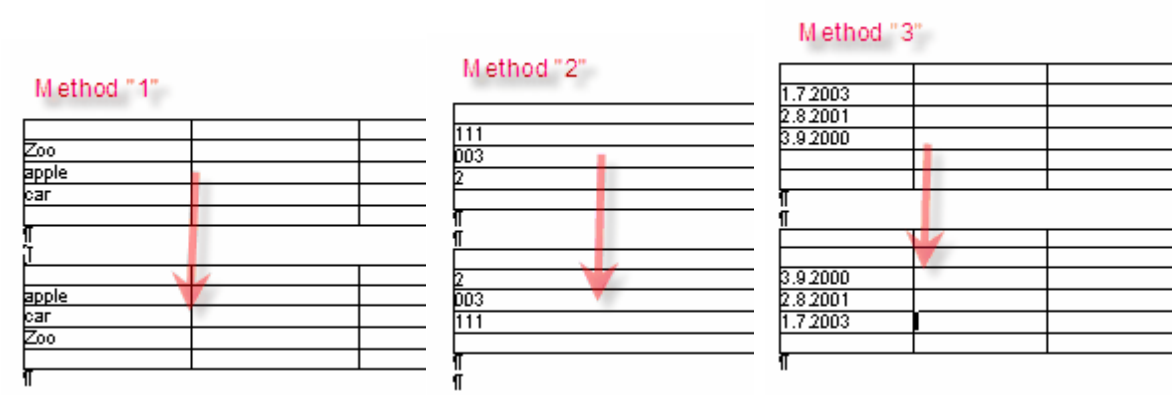
TextCursor.TableSort(1,1,0,0);

Header	
a111	
aaaa	
qweqw	
zzzzz	
Footer	

The following compare methods are supported:

0 : Direct, case sensitive compare

1 : Case insensitive
2 : Expect floating point number and compare
3 : Expect date value and compare



4.2.2.108 IWPTextCursor.ASetCellProp

Applies to
IWPTextCursor

Declaration
```
void ASetCellProp(int HeaderRows, bool OddRows, bool EvenRows, int FooterRows,
int FromColumn, int ToColumn, int WPAT_Code, int Value, int Mode);
```

Description
Applies a certain attribute to either header, footer or body rows. For table body rows you can select if odd and/or even rows should be changed.

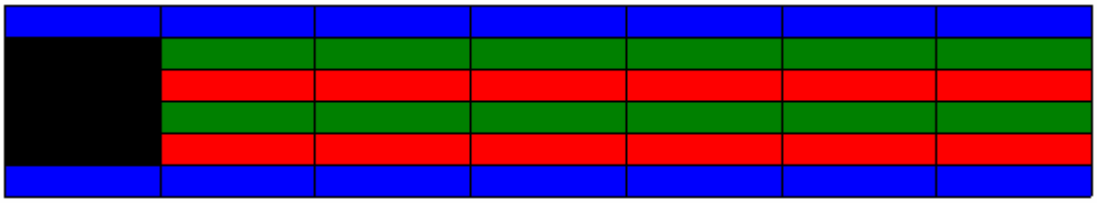
The value of HeaderRows or FooterRows must be the negative to leave this rows unchanged!

It is possible to only set a range (1...n) of columns or a complete rows with both FromColumn and TopColumn = 0.

The parameter Mode changes the way the attribute is applied:
-3 : clear all character attributes (from the text!)
-2 : clear all paragraph attributes
-1 : remove the given attribute
0 : set the attribute
1 : OR the attribute with the value
2 : AND NOT the attribute with the value
3 : Increase the value of the attribute by the given amount
4 : Set the row number as text
5 : Set the row number as literal A..Z as text
6 : Merges with previous row. (Not in first cell of the selected from-to range!)

Example:

```
// Odd rows should be green
TextCursor.ASetCellProp( -1, true, false, -1, 0, 0, (int)WPAT.FGColor, 2, 0);
// Even rows should be red
TextCursor.ASetCellProp( -1, false, true, -1, 0, 0, (int)WPAT.FGColor, 1, 0);
// Header and footer should be blue
TextCursor.ASetCellProp( 1, false, false, 1, 0, 0, (int)WPAT.FGColor, 3, 0);
// First column should be black
TextCursor.ASetCellProp( -1, true, true, -1, 1, 1, (int)WPAT.FGColor, 15, 0);
```



4.2.2.109 IWPTextCursor.ASetCellStyle

Applies to
IWPTextCursor

Declaration
```
void ASetCellStyle(int HeaderRows, [In] bool OddRows, bool EvenRows, int
FooterRows, int FromColumn, int ToColumn, string StyleName, int Mode);
```

Description
Applies a certain attribute to either header, footer or body rows. For table body rows you can select if odd and/or even rows should be changed.

The value of HeaderRows or FooterRows must be the negative to leave this rows unchanged!

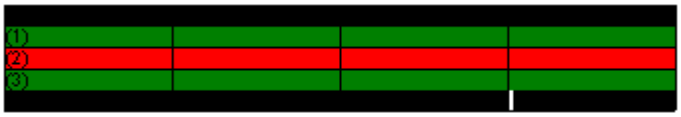
It is possible to only set a range (1...n) of columns or a complete rows with both FromColumn and ToColumn = 0.

If parameter Modes<33 it is used as bit-field:
0 : Select the paragraph style (adds it if necessary)
1 : Select the paragraph style and clear the paragraph attributes
2 : Select the paragraph style and clear the character attributes stored in the paragraph
4 : Select the paragraph style and clear the character attributes of the text!

Otherwise the following modes are supported:
33 : Initialize the cell text with the text provided as "StyleName"
34 : Append the text provided as "StyleName"
35 : Format the current row number according to the format string StyleName

Example:
```
// Odd rows should use style "td_odd"
TextCursor.ASetCellStyle( -1, true, false, -1, 0, 0, "td_odd", 0);
// Even rows should use style "td_even"
TextCursor.ASetCellStyle( -1, false, true, -1, 0, 0, "td_even", 0);
// Header and footer should use style "td_header"
TextCursor.ASetCellStyle( 1, false, false, 1, 0, 0, "td_header", 0);
// First column should be numbered (SetText)
TextCursor.ASetCellStyle( -1, true, true, -1, 1, 1, "(%d)", 35);

// Update the styles.
// Note: CurrStyle could be also used directly after ASetCellStyle!
Memo.SelectStyle("td_odd");
Memo.CurrStyle.ParASet((int)WPAT.FGColor, 2); // green
Memo.SelectStyle("td_even");
Memo.CurrStyle.ParASet((int)WPAT.FGColor, 1); // red
Memo.SelectStyle("td_header");
Memo.CurrStyle.ParASet((int)WPAT.FGColor, 15); // black
```

```
// Format and update
Memo.ReformatAll(false, true);
```



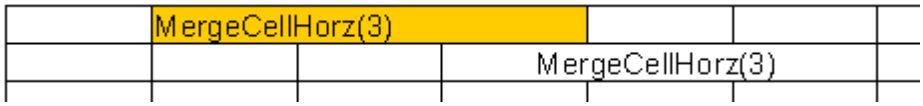
4.2.2.110 IWPTextCursor.MergeCellHorz

Applies to
IWPTextCursor

Declaration
```
bool MergeCellHorz(int CellCount);
```

Description
Merges the selected cells with CellCount = 0 (same effect as CombineCellHorz).

Otherwise, CellCount must be >1, the number of cells are merged:



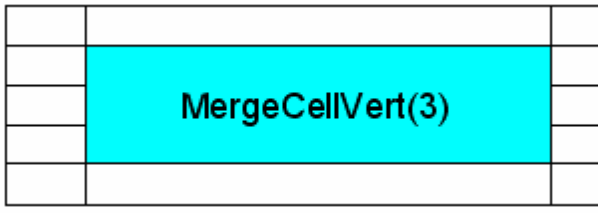
4.2.2.111 IWPTextCursor.MergeCellVert

Applies to
IWPTextCursor

Declaration
```
bool MergeCellVert(int CellCount);
```

Description
Merges the selected cells when CellCount = 0. In this case the method works like CombineCellVert.

Otherwise the number of Rows (CellCount must be >1) are vertically merged:



4.2.2.112 IWPTextCursor.ClearAllHeaders

Applies to
IWPTextCursor

Declaration
```
void ClearAllHeaders();
```

Description
Clears all headers in the text.

4.2.2.113 IWPTextCursor.ClearAllFooters

Applies to
IWPTextCursor

Declaration
```
void ClearAllFooters();
```

Description
Clears all footers in the text.

4.2.3 Example: Convert to HTML

This C# example loops through the text and create HTML which consist only of <p> and <a>

```
IWPMemo Memo = wpdllInt1.Memo;
StringBuilder HtmlText = new StringBuilder();
IWPTextCursor TextCursor;
IWPTextObj Obj;
IWPParInterface par = Memo.CurrPar;
bool parstart = true;

Memo = wpdllInt1.Memo;
TextCursor = Memo.TextCursor;
Obj = Memo.CurrObj;

TextCursor.GotoStart();
while(true)
{
      if (parstart)
      {
            HtmlText.Append("<p>");
            parstart = false;
      }

      if (TextCursor.CPObjPtr!=0)
      {
               if (Obj.ObjType==TextObjTypes.wpobjHyperlink)
               {
                 HtmlText.Append("<a href=\"");
                     HtmlText.Append(Obj.Command);
                     HtmlText.Append("\">");
                     HtmlText.Append(Obj.EmbeddedText);
                     HtmlText.Append("</a>");
                     Obj.MoveCursor(4); // Before end tag - we skip later
               }
      }
      else if (TextCursor.CPPosInPar>=par.CharCount)
      {
               HtmlText.Append("</p>\r\n");
               parstart = true;
      }
```

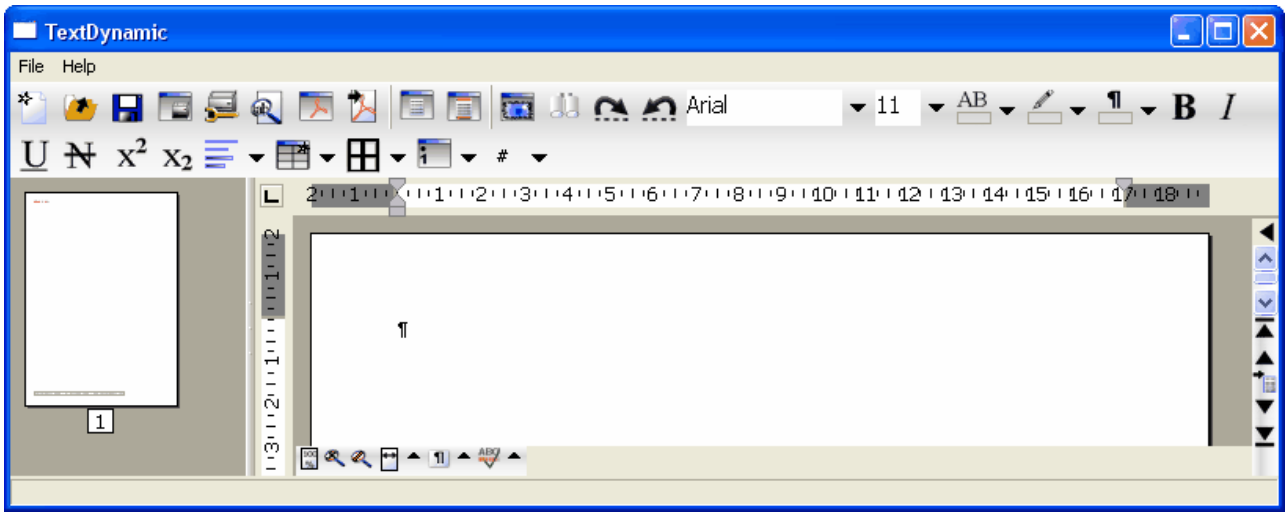
```
            else HtmlText.Append( (char)par.GetChar(TextCursor.CPPosInPar), 1 );
            if (!TextCursor.CPMoveNext()) break;
        }

        MessageBox.Show(HtmlText.ToString(),"");
```

4.3 TextDynamic as "Popup" or embedded editor (method wptextCreateDialog)

Do you have a C++Program which you need a powerful Editor for?
Or do you sell a database application which needs an new editing popup dialog.

With TextDynamic You can add a powerful editor with only a few lines of code.

When You have licensed WRTF2PDF/TextDynamic Server, You can license TextDynamic at a discount price.



To do so, the DLL exports a single procedure to create a complete editor window.

There are still various possibilities to customize the editor. Only the DLL is used in this case, no ActiveX or .NET wrapper.

This method has been created to allow editing fast, with a minimum programming interface.

4.3.1 Initialization of the editor in C++

Only a few lines of code are required to build a verstile and powerful editor in Visual Studio 2008:

Our demo uses this code to initialize the "create" function

```
// Pointer to TextDynamic
fkt_wptextCreateDialog* wptextCreateDialog;
```

#define TD_DLL "WPTextDLL01Demo.dll" ***MODIFY NAME***

```
// Load the DLL and get the method pointer
bool InitTD()
{
    static HINSTANCE Engine;
      Engine = LoadLibrary(TEXT(TD_DLL));
```

```
      if(Engine==NULL)return false;
    wptextCreateDialog = (fkt_wptextCreateDialog *)GetProcAddress( Engine,
"wptextCreateDialog" );
      if(wptextCreateDialog==NULL)return false;
      else return true;
}
```

Later, in

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
```

this code is executed:

1) Creation of editor

```
      case WM_CREATE:
            {
                  DefWindowProc(hWnd, message, wParam, lParam);
                  unsigned long editorw; //not used, reserved
                  static TDCreateDialogParam param;

                  if(InitTD())
                  {
                        param.size = sizeof(param);
                        param.modes = 1;
                        param.editor_mode = 4; // with thumbnails
                        param.editor_xmode = 255; // all features
                        param.editor_gui1 = 478;
                        param.editor1_layoutmode = 7; // Full Layout
                        param.editor1_autozoom = 1; // Width

                        // We can set all "Memo.SetBProp" flags.
                        // Using Group=0, ID=11 (bit 11) we activate Image
drag&drop
                        param.Memo1_SetBProp_ADD[0] = (1024 * 2); // Bit 11 ->
AcceptFiles

                        param.pccfile = "{dll}buttons.pcc";
                        edhwnd = wptextCreateDialog(hWnd, &param, 10, 10, 300,
300, &editorw );
            }
```

2) Sizing of the editor

```
      case WM_SIZE:
            DefWindowProc(hWnd, message, wParam, lParam);
            if(edhwnd)
            {
                  RECT Rect;
                  GetClientRect(hWnd, &Rect);
                  MoveWindow(edhwnd, Rect.left, Rect.top, Rect.right-Rect.left,
Rect.bottom-Rect.top, false);
            }
            break;
```

```
3) Avoid flickering

      case WM_ERASEBKGND: //<---

            break;

Thats All!
```

4.3.2 Use WPA Actions with Editor

Using the editor window handle, it is also possible to send commands to the editor using the window API SendMessage.

Very easy to use are the "wpa Actions".

Simply use this **message IDs** to send the **action names** to the current, the first or the second editor.

```
WMP_WPA_PROCESS        = 1051; // wparam= char *wpaAction, lparam=parama  - for c
WMP_WPA_PROCESS_1      = 1052; // wparam= char *wpaAction, lparam=parama  - for c
WMP_WPA_PROCESS_2      = 1053; // wparam= char *wpaAction, lparam=parama  - for c
```

You can use all "wpa" action names, such as "diaOpen", "diaSave" etc. See complete List.

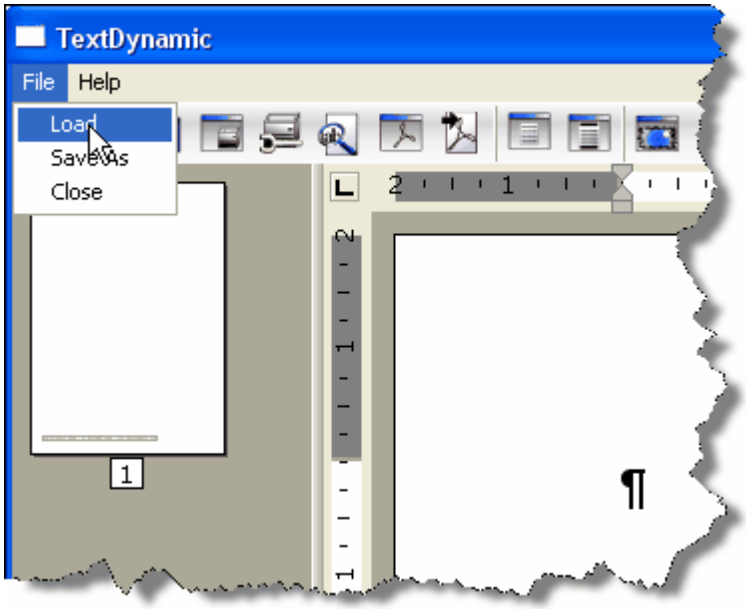
This pascal code will show the file open dialog:

```
SendMessage(EditorHwnd, 1051,  Integer(PChar('diaOpen')), 0);
```

the same in C++:

```
SendMessage(edhwnd, 1051, (int)((char *)"diaOpen"), 0);
```

In c++ You can add this code into the message loop to attach load and save menu items.




```
      case WM_COMMAND:
```

```
            wmId    = LOWORD(wParam);
            wmEvent = HIWORD(wParam);
            switch (wmId)
            {
            case ID_FILE_LOAD:
                 SendMessage(edhwnd, 1051, (int)((char *)"diaOpen"), 0);
                 break;
             case ID_FILE_SAVE:
                   SendMessage(edhwnd, 1051, (int)((char *)"diaSaveAs"),
0);
                 break;
```

4.3.3 Use Commands

Above we described how to use wpa Actions.

But You can also use one of the many commands which execute code directly.

These message IDs will be useful to send other commands

```
WMP_EDCOMMAND = 1040; // wparam=commandid, lparam=parameter for EDITOR A
WMP_EDbCOMMAND = 1041; // wparam=commandid, lparam=parameter for EDITOR B
WMP_EDCOMMANDSTR = 1042; // wparam=commandid, lparam=char* for EDITOR A
WMP_EDbCOMMANDSTR = 1043; // wparam=commandid, lparam=char* for EDITOR A
WMP_EDUCOMMANDSTR = 1044; // wparam=commandid, lparam=widechar* for EDITOR A
WMP_EDUbCOMMANDSTR = 1045; // wparam=commandid, lparam=widechar* for EDITOR B
```

For example You can use this command IDs to load and save directly to a file:

```
WPDLL_COM = 9000;
WPDLL_COM_TEXT_LOADFILE     = WPDLL_COM + 26; // buffer1=filename, buffer2=forma
WPDLL_COM_TEXT_SAVEFILE     = WPDLL_COM + 27; // buffer1=filename, buffer2=forma
WPDLL_COM_TEXT_LOADSTR      = WPDLL_COM + 28; // buffer1=str, buffer2=format, pa
WPDLL_COM_TEXT_SAVESTR      = WPDLL_COM + 29; // out: datastring, buffer2=forma
```

Example - load the file c:\test.rtf into the editor.

```
SendMessage(edhwnd, 1042, 9026,(int)((char *)"c:\\TEST.RTF"));
```

```
To work with a unicode (wide char) parameter use
```

```
SendMessage(edhwnd, 1044, 9026,(int)((widechar *)"c:\\TEST.RTF"));
```

```
//Set the inital directory for text and graphic
SendMessage(edhwnd, 1042, WPDLL_COM_INITIALDIR, (int)((char *)"c:\\temp"));

SendMessage(edhwnd, 1042, WPDLL_COM_INITIALGRAPHICDIR, (int)((char *)"c:\\temp"));


// Retrieve the current "InitialDir"
SendMessage(edhwnd, 1042, WPDLL_COM_INITIALDIR, 0);
```

4.3.4 Use the COM Interfaces

You can also use the COM interface IWPMemo.

This commands are used to retrieve a reference to the respective interface:

```
WPDLL_COM_Memo1              = 145;  // Result :=  IWPMemo of Editor 1
WPDLL_COM_Memo2              = 146;  // Result :=  IWPMemo of Editor 2
```

The definition of the Interfaces (TLB) is included in the TextDynamic OCX. You can import this TLB into Your project and autogenerate a the definition code.

4.3.5 Examples

Here we show how to access the embedded editor. The variable edhwnd is the window handle of the editor. It is returned by the method `fkt_wptextCreateDialog`. Please see Initialization of the editor in C++

Load the file c:\test.rtf into the editor. We use the command `WPDLL_COM_TEXT_LOADFILE` (See c++ header)

```
SendMessage(edhwnd, 1042, WPDLL_COM_TEXT_LOADFILE,(int)((char *)"c:\
\TEST.RTF"));
```

Select the language for the toolbar and dialogs:

```
SendMessage(edhwnd, 1042, WPSYS_SETLANGUAGE , (int)((char *)"DE")); // Select DE = German
```

Switch the tooltips on

```
SendMessage(edhwnd, WMP_EDCOMMAND, WPSYS_SETSHOWHINT, 1);
```

Set the initial text and image directory

```
SendMessage(edhwnd, 1042, WPDLL_COM_INITIALDIR, (int)((char *)"c:\\temp"));
SendMessage(edhwnd, 1042, WPDLL_COM_INITIALGRAPHICDIR, (int)((char *)"c:\\temp"
```

Retrieve and display the current directory

```
SendMessage(edhwnd, 1042, WPDLL_COM_INITIALDIR, 0);
int len;
char *p;
len = SendMessage(edhwnd,1042,WPDLL_DATA_GETSTRING, 0);
p = (char *)malloc(len);
SendMessage(edhwnd,1042,WPDLL_DATA_GETSTRING, (int)p);
MessageBoxA(0, p, "", 0);
free(p, len);
```

Show the file open dialog and display the file name of the loaded file in the caption

```
// Load file
SendMessage(edhwnd, 1051, (int)((char *)"diaOpen"), 0);
```

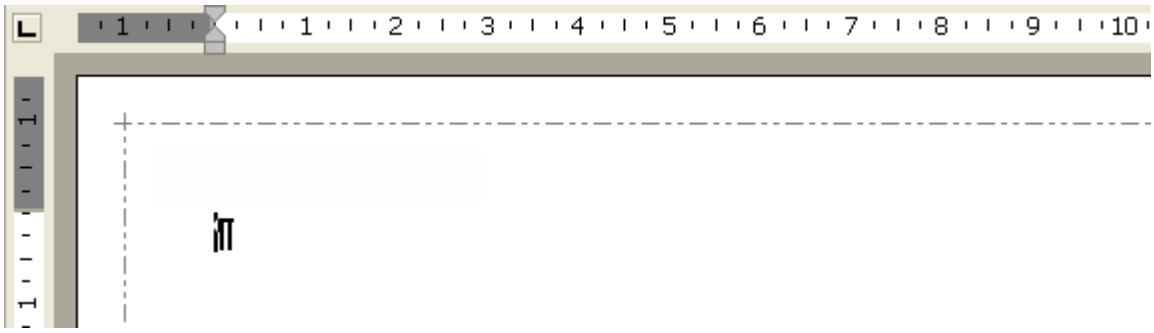
```
// Display file name
SendMessage(edhwnd, 1042, WPDLL_COM_LASTFILENAME, 0);
int len;
char buf[256];
len = SendMessage(edhwnd,1042,WPDLL_DATA_GETSTRINGW, 0);
if(len<256)
{
    SendMessage(edhwnd,1042,WPDLL_DATA_GETSTRINGW, (int)buf);
   SendMessage(hWnd, WM_SETEXT, 0, (int)&buf[0] );
}
```

Initialize the editor to use a page size of DinA 4 with margins of 1.5 cm:

```
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGE_WIDTH, (int)(21/2.54*1440)); // 21cm conv
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGE_HEIGHT, (int)(29.7/2.54*1440)); // 29.7cm
int margin = (int)(1.5/2.54*1440); // 1.5cm converted in twips
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGE_LEFT_MARGIN, margin);
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGE_TOP_MARGIN, margin);
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGE_RIGHT_MARGIN, margin);
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGE_BOTTOM_MARGIN, margin);
```

Display the non-printable area (0.5 cm on all sides).
Please see Memo.TextCommand(24) for other draw modes.

```
//Display frames to show not printable area
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGE_DRAW_MODE, 4); // Draw the frames
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGE_LINE_MODE, 2); // Draw lines
int offset = (int)(0.5/2.54*1440); // 1.5cm converted in twips
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGE_LINEOFFSET_LEFT, offset);
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGE_LINEOFFSET_TOP, offset);
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGE_LINEOFFSET_RIGHT, offset);
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGE_LINEOFFSET_BOTTOM, offset);
```



Set a certain fixed page count. The editor should always display 8 pages. Please also see Memo.TextCommand(24).

```
// Fix the page count
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGES_MINCOUNT, 4); // minimum 4 page
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGES_MAXCOUNT, 8); // maximum 8 page
```

Please note, that a second editor can show the same text with a different count of pages.
For example the thumbnail view will display less then 4 pages unless You use the code:

```
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED2_PAGES_MINCOUNT, 4); // minimum 4 page
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED2_PAGES_MAXCOUNT, 8); // maximum 8 page
```

It is also possible to force the page count to be dividable by 4:

```
SendMessage(edhwnd, WMP_SETIPROP, WIPR_ED1_PAGES_MULTIPLICATOR, 4);
```

4.3.6 Pascal Definition

```
unit WPDLL_CreateDlgInt;
// ----------------------------------------------------------------------------
// Create TextDynamic Interface
// Date: 5.10.2009
// ----------------------------------------------------------------------------

interface

uses Windows;

type

fkt_wptextDialogCB = function(
     handle : HWND;
     code   : Integer; // mesage id
     iparam : Integer; //
     lparam : Cardinal;
     sparam : PAnsiChar;
     memo   : Pointer // IWPMemo // can be null. Current editor
    ) : Integer;

    fkt_wptextDialogShow = function(
     handle : HWND;
     code   : Integer; // message id
     memo   : Pointer; // IWPMemo;  // can be null. First editor
     memo2   : Pointer // IWPMemo // can be null. Second editor
    ) : Integer;

  // AnsiChar version
  TDCreateDialogParam = packed record
   size     : Integer;
   // window
   modes          : Integer;
   // 1=show statusbar
   // 2=modal dialog
```

```
  toppanel_height: Integer;
  // 1=show status
  border         : Integer;
  // EditorStart  (or use wptextSetLicense)
  licensename: PAnsiChar;
  licensekey : PAnsiChar;

  // SetLayout(pccfile, pcclayout, pccpasswd,  "main", "main")
  //  pcclayout defaults to "default"
  //  pccpasswd defaults to ""
  pccfile : PAnsiChar;
  pcclayout : PAnsiChar;
  pccpasswd : PAnsiChar;

  // SetEditorMode
  editor_mode : Integer;
  editor_xmode : Integer;
  editor_gui1 : Integer;
  editor_gui2 : Integer;

  // Layoutmode and Zoom  (-1 = default, unchanged)
  editor1_layoutmode : Integer;
  editor1_autozoom : Integer;
  editor1_zoom : Integer;
  editor2_layoutmode : Integer;
  editor2_autozoom : Integer;
  editor2_zoom : Integer; // Ignored for thumbnailmode

  // Toolbar Colors
  toolbar_mode,  toolbar_colfrom, toolbar_colto : Cardinal;

  // Background Colors
  desktop_mode : Cardinal;
  desktop_colfrom : Cardinal;
  desktop_colto : Cardinal;

  // SetBProp - Index=Group, Bit Nr=Id, ADD to set, DEL to clear
  Memo1_SetBProp_ADD : array[0..19] of Cardinal;
  Memo1_SetBProp_DEL : array[0..19] of Cardinal;
  Memo2_SetBProp_ADD : array[0..19] of Cardinal;
  Memo2_SetBProp_DEL : array[0..19] of Cardinal;

  // Load a file
  loadfile : PAnsiChar;
  loadfilemode : Integer; // Reserved;

  // Callbacks
  msgwindow : HWND;
  OnQueryClose : fkt_wptextDialogCB;  // result=0 -> may not close
  AfterShow    : fkt_wptextDialogShow; // After show dialog - now can modify

  // Spellcheck
  spell_inifile : PChar;
  spell_langid  : Integer; // selected language
  spell_modes   : Integer; // Activate SpellAsYouGo ...

end;
```

```
  PTDCreateDialogParam = ^TDCreateDialogParam;

  fkt_wptextCreateDialog = function( parent : HWND; params : PTDCreateDialogParam; x,y,w,

const // MessagIDs to launch operation in editor
  WMP_DLLBASE = 1029; // WM_USER ...
  WPDLL_SYS  = 10;  // System commands- Use WMP_EDCOMMAND
  WPDLL_DATA = 100; // Create new data
  WPDLL_GUI = 1000; // GUI commands, enable, disable etc
  WPDLL_COM = 9000; // ..9999 Commands sent to the editor, change attributes etc
  WMP_EXCOMMAND = WMP_DLLBASE + 1; // lparam=ptr to TCOMEditDataCommand record, wparam=Si
  WMP_SETIPROP = WMP_DLLBASE + 2; // Set Integer Property: wparam=id, lparam = value

 WPDLL_DATA_GETSTRING     = WPDLL_DATA + 16; // Get Length of data buffer, fill buffer
 WPDLL_DATA_GETSTRINGW    = WPDLL_DATA + 17; // Get Length of data buffer, fill WIDECHAR

  WMP_EDCOMMAND = WMP_DLLBASE + 11; // wparam=commandid, lparam=parameter for EDITOR A
  WMP_EDbCOMMAND = WMP_DLLBASE + 12; // wparam=commandid, lparam=parameter for EDITOR B
  WMP_EDCOMMANDSTR = WMP_DLLBASE + 13; // wparam=commandid, lparam=char* for EDITOR A
  WMP_EDbCOMMANDSTR = WMP_DLLBASE + 14; // wparam=commandid, lparam=char* for EDITOR A
  WMP_EDUCOMMANDSTR = WMP_DLLBASE + 15; // wparam=commandid, lparam=widechar* for EDITOR
  WMP_EDUbCOMMANDSTR = WMP_DLLBASE + 16; // wparam=commandid, lparam=widechar* for EDITOR
  WMP_WPA_PROCESS       = WMP_DLLBASE + 22; // wparam= char *wpaAction, lparam=parama  -
  WMP_WPA_PROCESS_1     = WMP_DLLBASE + 23; // wparam= char *wpaAction, lparam=parama  -
  WMP_WPA_PROCESS_2     = WMP_DLLBASE + 24; // wparam= char *wpaAction, lparam=parama  -

 WPSYS_ABOUTBOX        = WPDLL_SYS + 16; // show about box, use WMP_EDCOMMAND
 WPSYS_CLEAR           = WPDLL_SYS + 17; // clears both editors
 WPSYS_SETLANGUAGE     = WPDLL_SYS + 18; // set language
 WPSYS_SETSPLITTERPOS  = WPDLL_SYS + 19; // set splitter pos in %
 WPSYS_GETSHOWHINT     = WPDLL_SYS + 20; // ShowHint
 WPSYS_SETSHOWHINT     = WPDLL_SYS + 21; //


 WPDLL_COM_REFORMAT = WPDLL_COM + 0; // Reformat in next idle phase, Param=1 at once, 2=a
  WPDLL_COM_ASET = WPDLL_COM + 1; // ASet()
  WPDLL_COM_AGET = WPDLL_COM + 2; // AGet()
  WPDLL_COM_ASETDEFTABFILL  = WPDLL_COM + 3; // ASet()
  WPDLL_COM_ASETDEFTABCOLOR = WPDLL_COM + 4; // uses color UParam
  WPDLL_COM_STYLE_CREATE    = WPDLL_COM + 5; // create a style and select it. If existing
  WPDLL_COM_STYLE_SELECT    = WPDLL_COM + 6; // selects/creates a style (to edit it)
  WPDLL_COM_STYLE_DELETE    = WPDLL_COM + 7; // deletes a style
  WPDLL_COM_STYLE_GETLIST   = WPDLL_COM + 8; // read comma separated list.
  // Use WPDLL_DATA_GETSTRING to read it!
  WPDLL_COM_STYLE_GETWPCSS  = WPDLL_COM + 9;  // Read WPCSS - one style
  WPDLL_COM_STYLE_SETWPCSS  = WPDLL_COM + 10; // Set WPCSS  - one style
  WPDLL_COM_STYLE_GETALLASSTR = WPDLL_COM + 11; // Read all as string (Param=Format: 0=WP
  WPDLL_COM_STYLE_SETALLASSTR = WPDLL_COM + 12; // Set all as string (Param=Format: 0=WPC
  WPDLL_COM_STYLE_SAVEFILE    = WPDLL_COM + 13; // Save all to file
  WPDLL_COM_STYLE_LOADFILE    = WPDLL_COM + 14; // Load all from file
  // MAIN EDIT COMMANDS - some are superseded ---------------------------------
  WPDLL_COM_TEXT_CLEAR        = WPDLL_COM + 15; // Clear the text, Param=1 "ClearEx", + b
  WPDLL_COM_TEXT_CLEARBODY    = WPDLL_COM + 16; // Clear only body, keep header+footer
  WPDLL_COM_TEXT_BACKUP       = WPDLL_COM + 17; // backup template, Bit1 to ignore page s
  WPDLL_COM_TEXT_RESTORE      = WPDLL_COM + 18; // restore template
```

```
WPDLL_COM_TEXT_HIDESEL      = WPDLL_COM + 19; // hide selection
WPDLL_COM_TEXT_SELECTALL    = WPDLL_COM + 20; // select all
WPDLL_COM_TEXT_BEGINENDPRINT= WPDLL_COM + 21; // BeginPrint/EndPrint (Param=Begin)
WPDLL_COM_TEXT_PRINT        = WPDLL_COM + 22; // print, optionla Param, Param2 to selec
WPDLL_COM_TEXT_SELECTPRINTER= WPDLL_COM + 23; // select printer 'name'
WPDLL_COM_TEXT_PRINTDIA     = WPDLL_COM + 24; // print dialog
WPDLL_COM_TEXT_DIALOG       = WPDLL_COM + 25; // show modal dialog (previe etc)
// DIALOG_....
WPDLL_COM_TEXT_LOADFILE     = WPDLL_COM + 26; // buffer1=filename, buffer2=format, para
WPDLL_COM_TEXT_SAVEFILE     = WPDLL_COM + 27; // buffer1=filename, buffer2=format, para
WPDLL_COM_TEXT_LOADSTR      = WPDLL_COM + 28; // buffer1=str, buffer2=format, param=1 f
WPDLL_COM_TEXT_SAVESTR      = WPDLL_COM + 29; // out: datastring, buffer2=format, param
WPDLL_COM_POS_SET          = WPDLL_COM + 30; // param=what, param2=value, buffer=opt va
  POSGOTO_POSITION   = 0; // Absolute character Position
  POSGOTO_PAR        = 1; // Absolute Paragraph Nr
  POSGOTO_POSINPAR   = 2; // Position in current paragraph
  POSGOTO_PAGE       = 3; // Absolute Page Nr
  POSGOTO_FIELD      = 4; // Field (buffer)
  POSGOTO_Bookmark   = 5; // Bookmark (buffer)
WPDLL_COM_POS_GET           = WPDLL_COM + 31; // get position
// --- POS_PARAM_ ....
WPDLL_COM_POS_SELECT        = WPDLL_COM + 32; // param=what, param2=value, buffer=opt v
// subset of POS_PARAM_
WPDLL_COM_MOVE              = WPDLL_COM + 33; // param=what move (select end), Param2=1
WPDLL_COM_DROP_MARKER       = WPDLL_COM + 34; // drop marker
WPDLL_COM_COLLECT_MARKER    = WPDLL_COM + 35; // collect marker (goto last pos)
// --- DATA_PARAM_ ....
WPDLL_COM_ACTIVATE          = WPDLL_COM + 36; // Makes sure the editor is active
WPDLL_COM_TEXT_LOADSTREAM   = WPDLL_COM + 37; // buffer1=ISTREAM, buffer2=format, param
WPDLL_COM_TEXT_SAVESTREAM   = WPDLL_COM + 38; // buffer1=ISTREAM, buffer2=format, param


WPDLL_COM_ProcessWPA        = WPDLL_COM + 56; // buffer1 = wpaname, buffer2=param, if b
WPDLL_COM_CheckWPA          = WPDLL_COM + 57; // buffer1 = wpaname, result: bit1=enable
WPDLL_COM_FindHeader        = WPDLL_COM + 58; // param=range, buffer=1 = name, Result =
WPDLL_COM_FindFooter        = WPDLL_COM + 59; // param=range, buffer=1 = name, Result =
WPDLL_COM_GetRTFVariable    = WPDLL_COM + 60; // Buffer=name, Result=String
WPDLL_COM_SetRTFVariable    = WPDLL_COM + 61; // Buffer=name, buffer2=texts

WPDLL_COM_TextCursor        = WPDLL_COM + 62; // Result :=  IWPTextCursor
WPDLL_COM_PDFCreator        = WPDLL_COM + 63; // Result :=  IWPPdfCreator
WPDLL_COM_TextAttr          = WPDLL_COM + 64; // Result :=  CurrSelAttr or CurrAttr
WPDLL_COM_GetIDWPA          = WPDLL_COM + 65; // Result :=  if for wpa action
WPDLL_COM_GetExceptionList  = WPDLL_COM + 66; // Result :=  list of recent exceptions
WPDLL_COM_GetMAPI           = WPDLL_COM + 67; // Result :=  IWPMapi Interface
WPDLL_COM_GetSPELL          = WPDLL_COM + 68; // Result :=  IWPSpell Interface
// Input Commands (if possible better use IWPTextCursor!)
WPDLL_COM_InputText         = WPDLL_COM + 70; // Param=width in %, negativ=absolute twi
WPDLL_COM_InputParagraph    = WPDLL_COM + 71; // Param=Mode, Buffer=StyleName
WPDLL_COM_InputSection      = WPDLL_COM + 72; // reserved , param=mode
WPDLL_COM_InputTable        = WPDLL_COM + 73; // buffer=name
WPDLL_COM_InputTableRowStart= WPDLL_COM + 74; // param=Mode
WPDLL_COM_InputTableCell    = WPDLL_COM + 75; // buffer=StyleName, buffer2=Text
WPDLL_COM_InputTableRowEnd  = WPDLL_COM + 76; // Don't forget!
WPDLL_COM_InputField        = WPDLL_COM + 77; // buffer=name, buffer2=text
```

```
WPDLL_COM_InputHyperLink    = WPDLL_COM + 78; // buffer=URL, buffer2=text  (wrap text i
WPDLL_COM_InputBookmark     = WPDLL_COM + 79; // buffer=name, buffer2=text (wrap text i
WPDLL_COM_InputImage        = WPDLL_COM + 80; // param=1=link, buffer=FileName
WPDLL_COM_InputClearTabstop = WPDLL_COM + 81; // clears all tabs
WPDLL_COM_InputTabstop      = WPDLL_COM + 82; // param=Value, param2=kind, param3=FillM
WPDLL_COM_InputObject       = WPDLL_COM + 83; // param=Typ, param2=mode, buffer=name, b
WPDLL_COM_InputHeader       = WPDLL_COM + 84; // param=Range, buffer=text (rtf, html)
WPDLL_COM_InputFooter       = WPDLL_COM + 85; // param=Range, buffer=text (rtf, html)
WPDLL_COM_InputFootnote     = WPDLL_COM + 86; // param=mode, buffer=text
WPDLL_COM_InputTextbox      = WPDLL_COM + 87; // param=width, buffer=text (rtf, html)
WPDLL_COM_InputPicture      = WPDLL_COM + 88; // param=w, param2=h + UParam = IPicture
WPDLL_COM_InputPictureStream= WPDLL_COM + 89; // param=w, param2=h + UParam = IStream i
// The next 3 also set the result string to the current value
WPDLL_COM_INITIALDIR        = WPDLL_COM + 90; // Set the initial directory for file ope
WPDLL_COM_INITIALGRAPHICDIR = WPDLL_COM + 91; // Set the initial image directory for im
WPDLL_COM_LASTFILENAME      = WPDLL_COM + 92; // Retrieve the filename of the last load

WPDLL_COM_DrawToHDC = WPDLL_COM + 108; // UParam=HDC, GUIElement=X, Param3=Y, Param=Wid
WPDLL_COM_DrawToBitmap = WPDLL_COM + 109; // UParam=HBitmap, GUIElement=X, Param3=Y, Pa
WPDLL_COM_SavePageAsMetafile     = WPDLL_COM + 110; // param=pagenr, param2=options, bu

WPDLL_COM_CreateReport   = WPDLL_COM + 111;
WPDLL_COM_GETINTERFACE   = WPDLL_COM + 112;
WPDLL_COM_GETXY   = WPDLL_COM + 113;

WPDLL_COM_PrintToDC     = WPDLL_COM + 114; // for IViewObjectExImpl::Draw
WPDLL_COM_PrintToDCGetH = WPDLL_COM + 115; // get height - in twips
WPDLL_COM_EvGetButton   = WPDLL_COM + 116; // Helper for VisualFoxPRO
WPDLL_COM_EvGetContents = WPDLL_COM + 117;
WPDLL_COM_EvGetBand     = WPDLL_COM + 118;

WPDLL_COM_PROTECTEDTEXT = WPDLL_COM + 119; // Set SecurityOptions and protect them
WPDLL_COM_ConvertTokens = WPDLL_COM + 120; // Convert Tokens to ReportTemplate

WPDLL_COM_PrintPDF      = WPDLL_COM + 121;   // same as PDFCreator Print 1. Optionally
WPDLL_COM_PrintPDF2      = WPDLL_COM + 122; // same as PDFCreator PrintSecond

// ids - modify Editor #1
   WIPR_ED1_PAGE_WIDTH = 1;
   WIPR_ED1_PAGE_HEIGHT = 2;
   WIPR_ED1_PAGE_LEFT_MARGIN = 3;
   WIPR_ED1_PAGE_RIGHT_MARGIN = 4;
   WIPR_ED1_PAGE_TOP_MARGIN = 5;
   WIPR_ED1_PAGE_BOTTOM_MARGIN = 6;
   WIPR_ED1_PAGE_HEADER_MARGIN = 7;
   WIPR_ED1_PAGE_FOOTER_MARGIN = 8;
   WIPR_ED1_PAGE_DRAW_MODE = 9; // Background image, address ...
   WIPR_ED1_PAGE_LINE_MODE = 10; // 0=adress top-left, 1=addres top right, 2=lineoffset
   WIPR_ED1_PAGE_LINEOFFSET_LEFT = 11;
   WIPR_ED1_PAGE_LINEOFFSET_RIGHT = 12;
   WIPR_ED1_PAGE_LINEOFFSET_TOP = 13;
   WIPR_ED1_PAGE_LINEOFFSET_BOTTOM = 14;
   WIPR_ED1_PAGES_MINCOUNT = 15;
   WIPR_ED1_PAGES_MAXCOUNT = 16;
   WIPR_ED1_PAGES_COUNT = 17;
   WIPR_ED1_PAGES_MULTIPLICATOR = 18;
```

```
    // for Editor#2
    WIPR_ED2_PAGE_WIDTH = 101;
    WIPR_ED2_PAGE_HEIGHT = 102;
    WIPR_ED2_PAGE_LEFT_MARGIN = 103;
    WIPR_ED2_PAGE_RIGHT_MARGIN = 104;
    WIPR_ED2_PAGE_TOP_MARGIN = 105;
    WIPR_ED2_PAGE_BOTTOM_MARGIN = 106;
    WIPR_ED2_PAGE_HEADER_MARGIN = 107;
    WIPR_ED2_PAGE_FOOTER_MARGIN = 108;
    WIPR_ED2_PAGE_DRAW_MODE = 109; // Background image, address ...
    WIPR_ED2_PAGE_LINE_MODE = 110; // 0=adress top-left, 1=addres top right, 2=lineoffse
    WIPR_ED2_PAGE_LINEOFFSET_LEFT = 111;
    WIPR_ED2_PAGE_LINEOFFSET_RIGHT = 112;
    WIPR_ED2_PAGE_LINEOFFSET_TOP = 113;
    WIPR_ED2_PAGE_LINEOFFSET_BOTTOM = 114;
    WIPR_ED2_PAGES_MINCOUNT = 115;
    WIPR_ED2_PAGES_MAXCOUNT = 116;
    WIPR_ED2_PAGES_COUNT = 117;
    WIPR_ED2_PAGES_MULTIPLICATOR = 118;


// IDs for WPDLL_COM_TEXT_DIALOG
DIALOG_Print=1;          // = wpaPrintDialog
DIALOG_PrinterSetup=2;   // = wpaPrinterSetup
DIALOG_Find=3;           // = wpaSearch
DIALOG_Replace=4;        // = wpaReplace
DIALOG_OPEN=5;           // = wpaOpen
DIALOG_SAVE=6;           // = wpaSave
DIALOG_SAVEAS=7;         // = wpaSaveAs
DIALOG_Preview=8;
DIALOG_PageProp=9;
DIALOG_PagePropPaperNames=10; // like PageProp but show paper names, too
DIALOG_SectionProp=11;  // reserved
DIALOG_ParagraphProp=12;
DIALOG_Tabstops=13;
DIALOG_Bullet=14;
DIALOG_BulletOutlines=15;  // Bullets and Numbers and Outlines
DIALOG_ParagraphBorder=16;
DIALOG_StyleSheet=17;
DIALOG_OneStyle=18;
DIALOG_Spellcheck=19;
DIALOG_SpellOptions=20;
DIALOG_ExportToWord=21;
DIALOG_ExportToPDF=22;
DIALOG_INSSymbol=23;
DIALOG_INSTable=24;
DIALOG_INSGRAPHIC=25;
DIALOG_INSGRAPHICLINK=26;
DIALOG_INSTextBox=27;
DIALOG_INSHyperlink=28;
DIALOG_INSBookmark=29;
DIALOG_INSFields=30;
DIALOG_ReportBands=31;
DIALOG_PDFProperties=32;
DIALOG_PrintLabels=33;
DIALOG_PrintBooklet=34;
```

```
DIALOG_DocInfo=35; // info + stats
DIALOG_Templates=36; // quick replacement
DIALOG_DocVariable=37; // caption=caption param=name  (RTFVariable!)
DIALOG_WPAbout=38; // TextDynamic about form
DIALOG_WPDebug=39; // Debug form with current paragraph attributes
DIALOG_MessageBox=40; // caption=caption param=text
DIALOG_PagePropPaperTray=41;
DIALOG_ManageFormulas=42;
DIALOG_ManageHeaderFooter=43;
DIALOG_FontSelect = 44; // Standard Font Dialog

implementation

end.
```

4.3.7 C / C++ Headerfile

```
// TDCreateDlg.h
// ------------------------------------------------------------------------
// Create TextDynamic Interface
// Date: 5.10.2009
// ------------------------------------------------------------------------


// For callbacks
typedef void (__stdcall fkt_wptextDialogCB)
     (unsigned long handle,
     long code, // mesage id
     long iparam, //
     unsigned long lparam,
     char *sparam,
     void *memo // IWPMemo // can be null. Current editor
     );

typedef void (__stdcall fkt_wptextDialogShow)
     (unsigned long handle,
     long code, // mesage id
     void *memo, // IWPMemo // can be null. Current editor
    void *memo2
     );


   // AnsiChar version
typedef struct
  {
   int size;
   // window
   int modes;
   // 1=show statusbar
   // 2=modal dialog
   int toppanel_height;
   // 1=show status
   int border;
   // EditorStart  (or use wptextSetLicense)
   char *licensename; //
   char *licensekey; // : PAnsiChar;
```

```
   // SetLayout(pccfile, pcclayout, pccpasswd,  "main", "main")
   //  pcclayout defaults to "default"
   //  pccpasswd defaults to ""
   char *pccfile; //,
   char *pcclayout; //,
   char *pccpasswd; // : PAnsiChar;

   // SetEditorMode
   int editor_mode; //,
   int editor_xmode; //,
   int editor_gui1; //,
   int editor_gui2; //

   // Layoutmode and Zoom  (-1 = default, unchanged)
   int editor1_layoutmode; //
   int editor1_autozoom; //
   int editor1_zoom; //
   int editor2_layoutmode; //
   int editor2_autozoom; //
   int editor2_zoom; // // Ignored for thumbnailmode

   // Toolbar Colors
   unsigned long toolbar_mode;
   unsigned long toolbar_colfrom;
   unsigned long toolbar_colto;

   // Background Colors
   unsigned long desktop_mode;
   unsigned long desktop_colfrom;
   unsigned long desktop_colto;

   // SetBProp - Index=Group, Bit Nr=Id, ADD to set, DEL to clear
   unsigned long Memo1_SetBProp_ADD[20];
   unsigned long Memo1_SetBProp_DEL[20];
   unsigned long Memo2_SetBProp_ADD[20];
   unsigned long Memo2_SetBProp_DEL[20];

   // Load a file
   char *loadfile;
   int  loadfilemode;

   // Callbacks
   unsigned long msgwindow;
   fkt_wptextDialogCB OnQueryClose;  // result=0 -> may not close
   fkt_wptextDialogShow AfterShow; // After show dialog - now can modify

   // Spellcheck
   char *spell_inifile;
   int spell_langid; // selected language
   int spell_modes; // Activate SpellAsYouGo ...
} TDCreateDialogParam;

typedef HWND (__stdcall fkt_wptextCreateDialog)
    ( HWND parent,
     TDCreateDialogParam *params,
```

```
     int x,
    int y,
    int w,
    int h,
    unsigned long *ed
     );


// Constants
#define  WMP_DLLBASE 1029 // WM_USER ...
#define  WPDLL_SYS   10  // System commands
#define  WPDLL_DATA  100 // Create new data
#define  WPDLL_GUI  1000 // GUI commands, enable, disable etc
#define  WPDLL_COM  9000 // ..9999 Commands sent to the editor, change attributes etc

#define    WMP_EXCOMMAND WMP_DLLBASE + 1  // lparam=ptr to TCOMEditDataCommand record, wp
#define    WMP_SETIPROP WMP_DLLBASE + 2  // Set Integer Property: wparam=id, lparam value
  // ids - modify Editor #1 (also see Memo.TextCommand)
#define     WIPR_ED1_PAGE_WIDTH 1
#define     WIPR_ED1_PAGE_HEIGHT 2
#define     WIPR_ED1_PAGE_LEFT_MARGIN 3
#define     WIPR_ED1_PAGE_RIGHT_MARGIN 4
#define    WIPR_ED1_PAGE_TOP_MARGIN 5
#define     WIPR_ED1_PAGE_BOTTOM_MARGIN 6
#define     WIPR_ED1_PAGE_HEADER_MARGIN 7
#define     WIPR_ED1_PAGE_FOOTER_MARGIN 8
#define     WIPR_ED1_PAGE_DRAW_MODE 9  // Background image, address ...
#define     WIPR_ED1_PAGE_LINE_MODE 10  // 0=adress top-left, 1=addres top right, 2=lineo
#define     WIPR_ED1_PAGE_LINEOFFSET_LEFT 11
#define     WIPR_ED1_PAGE_LINEOFFSET_RIGHT 12
#define     WIPR_ED1_PAGE_LINEOFFSET_TOP 13
#define     WIPR_ED1_PAGE_LINEOFFSET_BOTTOM 14
#define     WIPR_ED1_PAGES_MINCOUNT 15
#define     WIPR_ED1_PAGES_MAXCOUNT 16
#define     WIPR_ED1_PAGES_COUNT 17
#define     WIPR_ED1_PAGES_MULTIPLICATOR 18

     // for Editor#2
#define     WIPR_ED2_PAGE_WIDTH 101
#define     WIPR_ED2_PAGE_HEIGHT 102
#define     WIPR_ED2_PAGE_LEFT_MARGIN 103
#define     WIPR_ED2_PAGE_RIGHT_MARGIN 104
#define     WIPR_ED2_PAGE_TOP_MARGIN 105
#define     WIPR_ED2_PAGE_BOTTOM_MARGIN 106
#define     WIPR_ED2_PAGE_HEADER_MARGIN 107
#define     WIPR_ED2_PAGE_FOOTER_MARGIN 108
#define     WIPR_ED2_PAGE_DRAW_MODE 109  // Background image, address ...
#define     WIPR_ED2_PAGE_LINE_MODE 110  // 0=adress top-left, 1=addres top right, 2=line
#define     WIPR_ED2_PAGE_LINEOFFSET_LEFT 111
#define     WIPR_ED2_PAGE_LINEOFFSET_RIGHT 112
#define     WIPR_ED2_PAGE_LINEOFFSET_TOP 113
#define     WIPR_ED2_PAGE_LINEOFFSET_BOTTOM 114
#define     WIPR_ED2_PAGES_MINCOUNT 115
#define     WIPR_ED2_PAGES_MAXCOUNT 116
#define     WIPR_ED2_PAGES_COUNT 117
#define     WIPR_ED2_PAGES_MULTIPLICATOR 118
```

```
#define  WMP_EDCOMMAND WMP_DLLBASE + 11  // wparam=commandid, lparam=parameter for EDITOR
#define  WMP_EDbCOMMAND WMP_DLLBASE + 12  // wparam=commandid, lparam=parameter for EDITO
#define  WMP_EDCOMMANDSTR WMP_DLLBASE + 13  // wparam=commandid, lparam=char* for EDITOR
#define  WMP_EDbCOMMANDSTR WMP_DLLBASE + 14  // wparam=commandid, lparam=char* for EDITOR
#define  WMP_EDUCOMMANDSTR WMP_DLLBASE + 15  // wparam=commandid, lparam=widechar* for ED
#define  WMP_EDUbCOMMANDSTR WMP_DLLBASE + 16  // wparam=commandid, lparam=widechar* for E
#define  WMP_WPA_PROCESS       WMP_DLLBASE + 22  // wparam= char *wpaAction, lparam=param
#define  WMP_WPA_PROCESS_1     WMP_DLLBASE + 23  // wparam= char *wpaAction, lparam=param
#define  WMP_WPA_PROCESS_2     WMP_DLLBASE + 24  // wparam= char *wpaAction, lparam=param

#define   WPSYS_ABOUTBOX        WPDLL_SYS + 16 // show about box
#define   WPSYS_CLEAR           WPDLL_SYS + 17 // clears both editors
#define   WPSYS_SETLANGUAGE     WPDLL_SYS + 18 // set language
#define   WPSYS_SETSPLITTERPOS  WPDLL_SYS + 19 // set splitter pos in %
#define   WPSYS_GETSHOWHINT     WPDLL_SYS + 20 // ShowHint
#define   WPSYS_SETSHOWHINT    WPDLL_SYS + 21 //

#define WPDLL_COM_REFORMAT WPDLL_COM + 0  // Reformat in next idle phase, Param=1 at once
#define  WPDLL_COM_ASET WPDLL_COM + 1  // ASet()
#define  WPDLL_COM_AGET WPDLL_COM + 2  // AGet()
#define  WPDLL_COM_ASETDEFTABFILL  WPDLL_COM + 3  // ASet()
#define  WPDLL_COM_ASETDEFTABCOLOR WPDLL_COM + 4  // uses color UParam
#define  WPDLL_COM_STYLE_CREATE    WPDLL_COM + 5  // create a style and select it. If exi
#define  WPDLL_COM_STYLE_SELECT    WPDLL_COM + 6  // selects/creates a style (to edit it)
#define  WPDLL_COM_STYLE_DELETE    WPDLL_COM + 7  // deletes a style
#define  WPDLL_COM_STYLE_GETLIST   WPDLL_COM + 8  // read comma separated list.
  // Use WPDLL_DATA_GETSTRING to read it!
#define  WPDLL_COM_STYLE_GETWPCSS  WPDLL_COM + 9   // Read WPCSS - one style
#define  WPDLL_COM_STYLE_SETWPCSS  WPDLL_COM + 10  // Set WPCSS  - one style
#define  WPDLL_COM_STYLE_GETALLASSTR WPDLL_COM + 11  // Read all as string (Param=Format:
#define  WPDLL_COM_STYLE_SETALLASSTR WPDLL_COM + 12  // Set all as string (Param=Format: (
#define  WPDLL_COM_STYLE_SAVEFILE    WPDLL_COM + 13  // Save all to file
#define  WPDLL_COM_STYLE_LOADFILE    WPDLL_COM + 14  // Load all from file
  // MAIN EDIT COMMANDS - some are superseded ---------------------------------
#define  WPDLL_COM_TEXT_CLEAR        WPDLL_COM + 15  // Clear the text, Param=1 "ClearEx",
#define  WPDLL_COM_TEXT_CLEARBODY    WPDLL_COM + 16  // Clear only body, keep header+foot
#define  WPDLL_COM_TEXT_BACKUP       WPDLL_COM + 17  // backup template, Bit1 to ignore pa
#define  WPDLL_COM_TEXT_RESTORE      WPDLL_COM + 18  // restore template
#define  WPDLL_COM_TEXT_HIDESEL      WPDLL_COM + 19  // hide selection
#define  WPDLL_COM_TEXT_SELECTALL    WPDLL_COM + 20  // select all
#define  WPDLL_COM_TEXT_BEGINENDPRINT WPDLL_COM + 21  // BeginPrint/EndPrint (Param=Begin
#define  WPDLL_COM_TEXT_PRINT        WPDLL_COM + 22  // print, optionla Param, Param2 to
#define  WPDLL_COM_TEXT_SELECTPRINTER WPDLL_COM + 23  // select printer 'name'
#define  WPDLL_COM_TEXT_PRINTDIA     WPDLL_COM + 24  // print dialog
// Sow a DIALOG_....
#define  WPDLL_COM_TEXT_DIALOG       WPDLL_COM + 25  // show modal dialog (preview etc)

// Load and save commands
#define  WPDLL_COM_TEXT_LOADFILE     WPDLL_COM + 26  // buffer1=filename, buffer2=format,
#define  WPDLL_COM_TEXT_LOADSTR      WPDLL_COM + 28  // buffer1=str, buffer2=format, para
#define  WPDLL_COM_TEXT_SAVESTR      WPDLL_COM + 29  // out: datastring, buffer2=format,
#define  WPDLL_COM_POS_SET          WPDLL_COM + 30  // param=what, param2=value, buffer=o
#define    POSGOTO_POSITION   0  // Absolute character Position
#define    POSGOTO_PAR        1  // Absolute Paragraph Nr
#define    POSGOTO_POSINPAR   2  // Position in current paragraph
```

```
#define    POSGOTO_PAGE       3  // Absolute Page Nr
#define    POSGOTO_FIELD      4  // Field (buffer)
#define    POSGOTO_Bookmark   5  // Bookmark (buffer)
#define  WPDLL_COM_POS_GET           WPDLL_COM + 31  // get position
  // --- POS_PARAM_ ....
#define  WPDLL_COM_POS_SELECT        WPDLL_COM + 32  // param=what, param2=value, buffer=
  // subset of POS_PARAM_
#define  WPDLL_COM_MOVE              WPDLL_COM + 33  // param=what move (select end), Para
#define  WPDLL_COM_DROP_MARKER       WPDLL_COM + 34  // drop marker
#define  WPDLL_COM_COLLECT_MARKER    WPDLL_COM + 35  // collect marker (goto last pos)
  // --- DATA_PARAM_ ....
#define  WPDLL_COM_ACTIVATE          WPDLL_COM + 36  // Makes sure the editor is active
#define  WPDLL_COM_TEXT_LOADSTREAM   WPDLL_COM + 37  // buffer1=ISTREAM, buffer2=format, p
#define  WPDLL_COM_TEXT_SAVESTREAM   WPDLL_COM + 38  // buffer1=ISTREAM, buffer2=format, p


#define  WPDLL_COM_ProcessWPA        WPDLL_COM + 56  // buffer1 wpaname, buffer2=param, i
#define  WPDLL_COM_CheckWPA          WPDLL_COM + 57  // buffer1 wpaname, result: bit1=ena
#define  WPDLL_COM_FindHeader        WPDLL_COM + 58  // param=range, buffer=1 name, Resul
#define  WPDLL_COM_FindFooter        WPDLL_COM + 59  // param=range, buffer=1 name, Resul
#define  WPDLL_COM_GetRTFVariable    WPDLL_COM + 60  // Buffer=name, Result=String
#define  WPDLL_COM_SetRTFVariable    WPDLL_COM + 61  // Buffer=name, buffer2=texts

#define  WPDLL_COM_TextCursor        WPDLL_COM + 62  // Result :=  IWPTextCursor
#define  WPDLL_COM_PDFCreator        WPDLL_COM + 63  // Result :=  IWPPdfCreator
#define  WPDLL_COM_TextAttr          WPDLL_COM + 64  // Result :=  CurrSelAttr or CurrAtt
#define  WPDLL_COM_GetIDWPA          WPDLL_COM + 65  // Result :=  if for wpa action
#define  WPDLL_COM_GetExceptionList  WPDLL_COM + 66  // Result :=  list of recent excepti
#define  WPDLL_COM_GetMAPI           WPDLL_COM + 67  // Result :=  IWPMapi Interface
#define  WPDLL_COM_GetSPELL          WPDLL_COM + 68  // Result :=  IWPSpell Interface
  // Input Commands (if possible better use IWPTextCursor!)
#define  WPDLL_COM_InputText         WPDLL_COM + 70  // Param=width in %, negativ=absolut
#define  WPDLL_COM_InputParagraph    WPDLL_COM + 71  // Param=Mode, Buffer=StyleName
#define  WPDLL_COM_InputSection      WPDLL_COM + 72  // reserved , param=mode
#define  WPDLL_COM_InputTable        WPDLL_COM + 73  // buffer=name
#define  WPDLL_COM_InputTableRowStart WPDLL_COM + 74  // param=Mode
#define  WPDLL_COM_InputTableCell    WPDLL_COM + 75  // buffer=StyleName, buffer2=Text
#define  WPDLL_COM_InputTableRowEnd  WPDLL_COM + 76  // Don't forget!
#define  WPDLL_COM_InputField        WPDLL_COM + 77  // buffer=name, buffer2=text
#define  WPDLL_COM_InputHyperLink    WPDLL_COM + 78  // buffer=URL, buffer2=text  (wrap t
#define  WPDLL_COM_InputBookmark     WPDLL_COM + 79  // buffer=name, buffer2=text (wrap t
#define  WPDLL_COM_InputImage        WPDLL_COM + 80  // param=1=link, buffer=FileName
#define  WPDLL_COM_InputClearTabstop WPDLL_COM + 81  // clears all tabs
#define  WPDLL_COM_InputTabstop      WPDLL_COM + 82  // param=Value, param2=kind, param3=
#define  WPDLL_COM_InputObject       WPDLL_COM + 83  // param=Typ, param2=mode, buffer=na
#define  WPDLL_COM_InputHeader       WPDLL_COM + 84  // param=Range, buffer=text (rtf, ht
#define  WPDLL_COM_InputFooter       WPDLL_COM + 85  // param=Range, buffer=text (rtf, ht
#define  WPDLL_COM_InputFootnote     WPDLL_COM + 86  // param=mode, buffer=text
#define  WPDLL_COM_InputTextbox      WPDLL_COM + 87  // param=width, buffer=text (rtf, ht
#define  WPDLL_COM_InputPicture      WPDLL_COM + 88  // param=w, param2=h + UParam IPictu
#define  WPDLL_COM_InputPictureStream WPDLL_COM + 89  // param=w, param2=h + UParam IStrea

// The following ignore an empty string. They always set the current result string to the
// Use WPDLL_DATA_GETSTRING to retrive the current string
#define  WPDLL_COM_INITIALDIR        WPDLL_COM + 90 // Set the initial directory for fil
```

```
#define  WPDLL_COM_INITIALGRAPHICDIR  WPDLL_COM + 91 // Set the initial image directory fo
#define  WPDLL_COM_LASTFILENAME  WPDLL_COM + 92 // Set / retrieve the current file name

#define  WPDLL_COM_DrawToHDC WPDLL_COM + 108  // UParam=HDC, GUIElement=X, Param3=Y, Para
#define  WPDLL_COM_DrawToBitmap WPDLL_COM + 109  // UParam=HBitmap, GUIElement=X, Param3=
#define  WPDLL_COM_SavePageAsMetafile     WPDLL_COM + 110  // param=pagenr, param2=option

#define  WPDLL_COM_CreateReport   WPDLL_COM + 111
#define  WPDLL_COM_GETINTERFACE   WPDLL_COM + 112
#define  WPDLL_COM_GETXY   WPDLL_COM + 113

#define  WPDLL_COM_PrintToDC     WPDLL_COM + 114  // for IViewObjectExImpl::Draw
#define  WPDLL_COM_PrintToDCGetH WPDLL_COM + 115  // get height - in twips
#define  WPDLL_COM_EvGetButton   WPDLL_COM + 116  // Helper for VisualFoxPRO
#define  WPDLL_COM_EvGetContents WPDLL_COM + 117
#define  WPDLL_COM_EvGetBand     WPDLL_COM + 118

#define  WPDLL_COM_PROTECTEDTEXT WPDLL_COM + 119  // Set SecurityOptions and protect them
#define WPDLL_COM_ConvertTokens WPDLL_COM + 120  // Convert Tokens to ReportTemplate

#define  WPDLL_COM_PrintPDF      WPDLL_COM + 121    // same as PDFCreator Print 1. Option
#define  WPDLL_COM_PrintPDF2      WPDLL_COM + 122  // same as PDFCreator PrintSecond

// Some commands write an ansi string. This command is used to retrieve the result string
// Call it with a null buffer to get the size, call it again with a correctly dimensioned
#define WPDLL_DATA_GETSTRING  WPDLL_DATA+ 16 // Get Length of data buffer, fill buffer
#define WPDLL_DATA_GETSTRINGW WPDLL_DATA+ 17  // Get Length of data buffer, fill WIDECHAR

// IDs for WPDLL_COM_TEXT_DIALOG
#define DIALOG_Print  1            //    wpaPrintDialog
#define DIALOG_PrinterSetup  2     //    wpaPrinterSetup
#define DIALOG_Find  3             //    wpaSearch
#define DIALOG_Replace  4          //    wpaReplace
#define DIALOG_OPEN  5             //    wpaOpen
#define DIALOG_SAVE  6             //    wpaSave
#define DIALOG_SAVEAS  7           //    wpaSaveAs
#define DIALOG_Preview  8
#define DIALOG_PageProp  9
#define DIALOG_PagePropPaperNames  10   // like PageProp but show paper names, too
#define DIALOG_SectionProp  11    // reserved
#define DIALOG_ParagraphProp  12
#define DIALOG_Tabstops  13
#define DIALOG_Bullet  14
#define DIALOG_BulletOutlines  15    // Bullets and Numbers and Outlines
#define DIALOG_ParagraphBorder  16
#define DIALOG_StyleSheet  17
#define DIALOG_OneStyle  18
#define DIALOG_Spellcheck  19
#define DIALOG_SpellOptions  20
#define DIALOG_ExportToWord  21
#define DIALOG_ExportToPDF  22
#define DIALOG_INSSymbol  23
#define DIALOG_INSTable  24
#define DIALOG_INSGRAPHIC  25
#define DIALOG_INSGRAPHICLINK  26
#define DIALOG_INSTextBox  27
```

```
#define DIALOG_INSHyperlink  28
#define DIALOG_INSBookmark  29
#define DIALOG_INSFields  30
#define DIALOG_ReportBands  31
#define DIALOG_PDFProperties  32
#define DIALOG_PrintLabels  33
#define DIALOG_PrintBooklet  34
#define DIALOG_DocInfo  35   // info + stats
#define DIALOG_Templates  36   // quick replacement
#define DIALOG_DocVariable  37   // caption  caption param  name  (RTFVariable!)
#define DIALOG_WPAbout  38   // TextDynamic about form
#define DIALOG_WPDebug  39   // Debug form with current paragraph attributes
#define DIALOG_MessageBox  40   // caption  caption param  text
#define DIALOG_PagePropPaperTray  41
#define DIALOG_ManageFormulas  42
#define DIALOG_ManageHeaderFooter  43
#define DIALOG_FontSelect    44   // Standard Font Dialog
```

4.4 Text Attributes (access and modify)

The interface IWPAttrInterface is published by property CurrAttr, TextAttr and CurrSelAttr. CurrAttr changes the current writing mode, CurrSelAttr the selected text and TextAttr provides automatically the CurrAttr or CurrSelAttr if text is selected.

IWPCharacterAttr is used to change the appearence of special text objects, such as the field markers.

If you need to modify paragraph attributes, such as indent, tab stops, shading please use the interface IWPParInterface which is provided by Memo.CurrPar.

4.4.1 IWPAttrInterface

This important interface is used to update the character attributes and paragraph attributes.

The interface is published by property CurrAttr, TextAttr and CurrSelAttr. CurrAttr changes the current writing mode, CurrSelAttr the selected text and TextAttr provides automatically the CurrAttr or CurrSelAttr if text is selected.
You can use TextAttr to fill your own toolbar with life, in case you do not want to use the inbuilt toolbar to change the attributes of the text. You can also use it to implement hotkeys, for example toggle 'bold' when pressing Ctrl+B.

The default text attributes can be changed in Memo.DefaultAttr. (If a function of TextAttr returns FALSE this means the default attribute is used for this property.)

The instance "AttrHelper" can be used to calculate CharAttrIndex values to be used with TextCursor.InputString.

A different instance is published as property CurrParAttr (Memo.CurrParAttr or RTFDataBlock.CurrParAttr). Here **any change will affect all characters in this paragraph**. Please note that it will not change the properties stored by the paragraph itself, but the properties used by the individual characters - in the attribute cache.

If you need to change the character attributes stored in the paragraph object you can use the method CurrPar.ParASet.

CurrStyleAttr is used to change the character attributes defined by a style. You can use SelectStyle to create a new paragraph style or select an existing style to be modified. To apply a style a style to a paragraph, you can use the property StyleName.

The interface IWPAttrInterface has several methods to read and write properties.

The basic methods which can access all attributes are AttrGet, AttrDel and AttrSet. These methods expect the property ID (WPAT...) - Please see the list of the available WPAT codes. If the AttrGet function returns the value false, the value has not been defined. This means the inherited or the default value will be used for this property. This three basic functions will modify the character attributes for IDs <=16, for IDs >16 the current paragraph / selected paragraphs will be modified. You can use Memo.TextAttr.AttrSET((int)WPAT.IndentLeft, 360) to change the indent of the current paragraph or all selected paragraphs, in case text was selected.

The other methods only affect the character attributes. For other changes on paragraphs or styles use IWPParInterface.

The property **CharAttrIndex** represents the writing mode as index into the attribute cache. The value can be used with TextCursor.InputText("some text", x). You can store it into an integer variable if you need to temporarily change the writing mode. When you are done simply assign the value and you can use the previous writing mode. Please note that clearing the text makes the index values invalid!

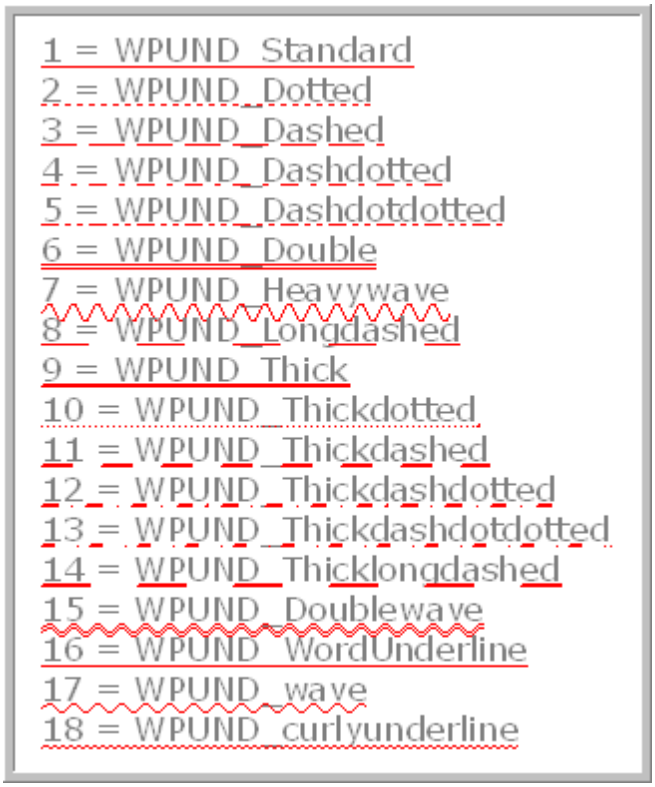
Please note: The methods to read a property "Get...." all return a boolean value. If this value is FALSE, this means the property is not used, the property uses the default value.

With .NET use the API ColorTranslator.FromWin32(rgb) to convert a RGB value into the Color type.

You can use ColorToRGB to convert a Color type into RGB.

Methods	Methods
AttrGet	**IncludeStyles**
AttrSet	**NrToColor**
BeginUpdate	**SetBGColor**
Clear	**SetBGColorNr**
ColorToNr	**SetCharStyleSheetName**
EndUpdate	**SetCharWidth**
ExcludeStyles	**SetColor** (changes the ForeColor)
GetBGColor	**SetColorNr**
GetBGColorNr	**SetFontCharset**
GetCharStyleSheetName	**SetFontface**
GetCharWidth	**SetFontSize**
GetColor	**SetStyles**
GetColorNr	**SetTextLanguage**
GetFontCharset	**SetUnderlineColor**
GetFontface	**SetUnderlineColorNr**
GetFontSize	**SetUnderlineMode**
GetStyles	**SetWPCSS**
GetTextLanguage	**ToggleStyle**
GetUnderlineColor	
GetUnderlineColorNr	
GetUnderlineMode	
GetWPCSS	

IWPAttrInterface Example



This C# example code creates text to show the possible underline modes.

```
 string[] cnames =  new string[]
[OBJECT]
 { "WPUND_Standard"
[OBJECT]
 , "WPUND_Dotted"
 , "WPUND_Dashed"
 , "WPUND_Dashdotted"
 , "WPUND_Dashdotdotted"
 , "WPUND_Double"
 , "WPUND_Heavywave"
 , "WPUND_Longdashed"
 , "WPUND_Thick"
 , "WPUND_Thickdotted"
 , "WPUND_Thickdashed"
 , "WPUND_Thickdashdotted"
 , "WPUND_Thickdashdotdotted"
 , "WPUND_Thicklongdashed"
 , "WPUND_Doublewave"
 , "WPUND_WordUnderline"
 , "WPUND_wave"
 , "WPUND_curlyunderline" };
 IWPMemo Memo = WPDLLInt1.Memo;
 IWPAttrInterface CurrAttr = Memo.CurrAttr;
 IWPTextCursor TextCursor = Memo.TextCursor;
 Memo.Clear(false,false);
 CurrAttr.Clear();
 CurrAttr.SetFontface("Verdana");
 //150% Line Height
```

```
CurrAttr.AttrSET((int)WPAT.LineHeight, 150);
int m = 1;
foreach (string s in cnames)
{
 TextCursor.InputParagraph(0, "");
 CurrAttr.SetUnderlineMode(m);
 CurrAttr.SetUnderlineColor(WPDLLInt1.ToRGB(Color.Red));
 TextCursor.InputText(m.ToString());
 TextCursor.InputText(" = ");
 TextCursor.InputText(s);
 m++;
}
Memo.Reformat();
```

4.4.1.1 Properties

4.4.1.1 A) CharAttrIndex

Index of the attribute record which holds character properties.

Applies to
IWPAttrInterface

Declaration
```
int CharAttrIndex;
```

Description
The TextDynamic word processing engine stores the attributes of characters in attribute records. This records hold 16 different attributes.

To reduce memory consumption the text only contains the reference, the index value, of the attribute record.

This property can be read and written. After you have assigned an index value you can use methods such as GetFontFace to read the actually used property elements. Methods such as InsertText can use such index values, as well. This makes it possible, if you intend to create text in program code, to first create a set of index values for different parts of the text and then use this pre calculated index values when creating the text.
Please note that the Attribute array is cleared when the the text is cleared.

The interface IWPAttrInterface can be also used to manipulate the current style In this case it is not possible to read CharAttrIndex. The interface is only used as helper to set font attributes, a index into the attribute cache is not created in this case.

This interface is also used to manipulate a "current" paragraph (Memo.CurrPar or RTFDataBlock.CurrPar). In this case reading CharAttrIndex will return the value which was applied last, it is the "default" attribute for a paragraph .

Category
Character Attributes

4.4.1.1 B) ID

This is the identification number of this style.

In case of number styles use it with WPAT_NumberStyle

```
IWPAttrInterface Attr = Memo.TextAttr;
IWPNumberStyle style;
style = Memo.GetNumberStyle(-1,1,0); // Bullet
```

```
if(style!=null)
{
    style.TextA = "¢"; // after text = "."
    style.Font="WingDings";
    style.Indent = 360;
}
Attr.AttrSET((int)WPAT.NumberSTYLE, style.ID);
```

4.4.1.2 Methods

4.4.1.2 A) IWPAttrInterface.AttrSet

Applies to
IWPAttrInterface

Declaration
```
procedure AttrSet(WPAT_Code: Integer; Value: Integer);
```

Description
Sets a certain property.

The property is selected by the WPAT code.

Note: Internally the properties are stored in an array, not in specific variables. Only if a property is defined, an entry is added to this array. This does not only save resources, but it allows all properties to have an undefined state. This feature is important for a clean support of paragraph styles.

The three basic functions (AttrSet, AttrGet, AttrDel) will modify the character attributes for IDs <=16, for IDs >16 the current paragraph / selected paragraphs will be modified. You can use Memo.TextAttr.AttrSET((int)WPAT.IndentLeft, 360) to change the indent of the current paragraph or all selected paragraphs, in case text was selected. But usually You will use CurrPar.ParASet in the same way.

4.4.1.2 B) IWPAttrInterface.AttrGet

Applies to
IWPAttrInterface

Declaration
```
function AttrGet(WPAT_Code: Integer; var Value: Integer): Boolean;
```

Description
Reads a certain property.

If the AttrGet function returns the value false, the value has not been defined - in this case the default value is used.

4.4.1.2 C) IWPAttrInterface.AttrDel

Applies to
IWPAttrInterface

Declaration
```
procedure AttrDel(WPAT_Code: Integer);
```

Description
Removes a certain property definition, the inherited or the default value will be used for this property.

4.4.1.2 D) IWPAttrInterface.BeginUpdate

Applies to
IWPAttrInterface

Declaration
```
procedure BeginUpdate;
```

Description
If you change multiply properties you can apply the changes at once if your first use BeginUpdate and EndUpdate when you are done. You can increaese the performance by doing so if the change affects the selected text.

4.4.1.2 E) IWPAttrInterface.Clear

Applies to
IWPAttrInterface

Declaration
```
void Clear()
```

Description
This method clears all properties - the default attributes or the inherited values (from a style or the current paragraph) will be then used for the text.

See SelectStyle for an example.

4.4.1.2 F) IWPAttrInterface.ColorToNr

Applies to
IWPAttrInterface

Declaration
```
int ColorToNr(string ColorName);
```

Description
This method can be used to convert a color value defined by a string (same syntax as used by CSS) to a number which can be used by AttrSet or SetBGColorNr and SetColorNr.

Also see IWPAttrInterface.NrToColor which converts a color id into a string describing the color.

Example:

```
Memo.CurrAttr.SetFontSize(8);
Memo.CurrAttr.SetColorNr( Memo.CurrAttr.ColorToNr("red") );
TextCursor.InputText("Some red text");
Memo.CurrAttr.SetColorNr(-1);
```

4.4.1.2 G) IWPAttrInterface.EndUpdate

Applies to
IWPAttrInterface

Declaration
```
function EndUpdate: Boolean;
```

Description
This method has to be used after BeginUpdate.

4.4.1.2 H) IWPAttrInterface.IncludeStyles

Applies to
IWPAttrInterface

Declaration
```
procedure IncludeStyles(Element: WrtStyle);
```

Description
This method adds a certain character style flag.

Parameters	
Element	**0 :** Bold text. (C# wrapper defines enum element WPWRT.BOLD) **1 :** Italic text. (C# wrapper defines enum element WPWRT.ITALIC) **2 :** Underlined text. (C# wrapper defines enum element WPWRT.UNDERLINE) **3 :** Strikeout text. (C# wrapper defines enum element WPWRT.STRIKEOUT) **4 :** Text in super-script (C# wrapper defines enum element WPWRT.SUPERSCRIPT) **5 :** Text in sub-script (C# wrapper defines enum element WPWRT.SUBSCRIPT) **6 :** Hidden text, (C# wrapper: WPWRT.HIDDEN) **7 :** Uppercase text. (C# wrapper: WPWRT.UPPERCASE) **8 :** reserved. **9 :** Lowercase text. (C# wrapper: WPWRT.LOWERCASE) **10 :** Text which should be excluded from spellcheck (WPWRT.NOPROOF) **11 :** Double strikeout (WPWRT.DBLSTRIKEOUT) **12 :** reserved. **13 :** protected text (WPWRT.PROTECTED)

Category
Character Styles

4.4.1.2 I) IWPAttrInterface.ExcludeStyles

Applies to
IWPAttrInterface

Declaration
```
procedure ExcludeStyles(Element: WrtStyle);
```

Description
This method is used to remove a certain character style flag.

Parameters	
Element	**0 :** Bold text. (C# wrapper defines enum element WPWRT.BOLD) **1 :** Italic text. (C# wrapper defines enum element WPWRT.ITALIC) **2 :** Underlined text. (C# wrapper defines enum element WPWRT.UNDERLINE) **3 :** Strikeout text. (C# wrapper defines enum element WPWRT.STRIKEOUT) **4 :** Text in super-script (C# wrapper defines enum element WPWRT.SUPERSCRIPT)

5 : Text in sub-script (C# wrapper defines enum element WPWRT.SUBSCRIPT)
6 : Hidden text, (C# wrapper: WPWRT.HIDDEN)
7 : Uppercase text. (C# wrapper: WPWRT.UPPERCASE)
8 : reserved.
9 : Lowercase text. (C# wrapper: WPWRT.LOWERCASE)
10 : Text which should be excluded from spellcheck (WPWRT.NOPROOF)
11 : Double strikeout (WPWRT.DBLSTRIKEOUT)
12 : reserved.
13 : protected text (WPWRT.PROTECTED)

Category
Character Styles

4.4.1.2 J) IWPAttrInterface.GetBGColor

Applies to
IWPAttrInterface

Declaration
```
bool GetBGColor(ref int Color);
```

Description
The method reads the background color used for text as RBG value.

In .NET use the API ColorTranslator.FromWin32(rgb) to convert a RGB value into the Color type.

4.4.1.2 K) IWPAttrInterface.GetBGColorNr

Applies to
IWPAttrInterface

Declaration
```
function GetBGColorNr(var ColorNr: Integer): Boolean;
```

Description
The method reads the background color used for text as index value.

4.4.1.2 L) IWPAttrInterface.GetCharStyleSheetName

Applies to
IWPAttrInterface

Declaration
```
function GetCharStyleSheetName(var StyleName: WideString): Boolean;
```

Description
The method reads the paragraph style name which is attached to the characters. Please do not mix up with StyleName which is used to attach a style to a paragraph.

4.4.1.2 M) IWPAttrInterface.GetCharWidth

Applies to
IWPAttrInterface

Declaration
```
function GetCharWidth(var CharWidthPC: Integer): Boolean;
```

Description
If the text uses compressed text, this method will read the width property.

4.4.1.2 N) IWPAttrInterface.GetColor / ForeColor

Applies to
IWPAttrInterface

Declaration
```
bool GetColor(ref int Color);
```

Description
This function reads the current character color (=ForeColor).

Please note:
The methods to read a property "Get...." all return a boolean value. #
If this value is FALSE, this means the property is not used, the property uses the default value.

In .NET applications you can use the utility function **ToRGB** to convert a *Color* member into a RGB value. Example:

```
WPDLLInt1.ToRGB(Color.Red)
```

Also see IWPAttrInterface.SetColor / ForeColor

4.4.1.2 O) IWPAttrInterface.GetColorNr

Applies to
IWPAttrInterface

Declaration
```
function GetColorNr(var ColorNr: Integer): Boolean;
```

Description
This function reads the current character color as index value.

It can be used with IWPAttrInterface.NrToColor.

4.4.1.2 P) IWPAttrInterface.GetFontCharset

Applies to
IWPAttrInterface

Declaration
```
function GetFontCharset(var Charset: Integer): Boolean;
```

Description
This function reads the charset used by the text.

4.4.1.2 Q) IWPAttrInterface.GetFontface

Applies to
IWPAttrInterface

Declaration
```
function GetFontface(var FontName: WideString): Boolean;
```

Description
This method reads the font name used for the text.

Please note:
The methods to read a property "Get...." all return a boolean value.
If this value is FALSE, this means the property is not used, the property uses the default value.

4.4.1.2 R) IWPAttrInterface.GetFontSize

Applies to
IWPAttrInterface

Declaration
```
function GetFontSize(var FontSize: Single): Boolean;
```

Description
This function reads the size used by the text. The parameter is a 32 bit floating point value.

4.4.1.2 S) IWPAttrInterface.GetStyles

Applies to
IWPAttrInterface

Declaration
```
function GetStyles(var Styles: WrtStyle): Boolean;
```

Description
This method retrieves the character style flags which are used.

Returns
this bitfield:
1 : Bold text. (C# wrapper defines enum element WPSTY.BOLD)
2 : Italic text. (C# wrapper defines enum element WPSTY.ITALIC)
4 : Underlined text. (C# wrapper defines enum element WPSTY.UNDERLINE)
8 : Strikeout text. (C# wrapper defines enum element WPSTY.STRIKEOUT)
16 : Text in super-script (C# wrapper defines enum element WPSTY.SUPERSCRIPT)
32 : Text in sub-script (C# wrapper defines enum element WPSTY.SUBSCRIPT)
64 : Hidden text, (C# wrapper: WPYST.HIDDEN)
128 : Uppercase text. (C# wrapper: WPYST.UPPERCASE)
256 : reserved.
512 : Lowercase text. (C# wrapper: WPYST.LOWERCASE)
1024 : Text which should be excluded from spellcheck (WPSTY.NOPROOF)
2048 : Double strikeout (WPSTY.DBLSTRIKEOUT)
4096 : reserved.
8192 : protected text (WPSTY.PROTECTED)

Category
Character Styles

4.4.1.2 T) IWPAttrInterface.GetTextLanguage

Applies to
IWPAttrInterface

Declaration
```
function GetTextLanguage(var Mode: Integer): Boolean;
```

Description
This method reads the language id used with this text. It is currently not used.

4.4.1.2 U) IWPAttrInterface.GetUnderlineColor

Applies to
IWPAttrInterface

Declaration
```
function GetUnderlineColor(var RGB: TColor): Boolean;
```

Description
This method reads the color used for underlines.

4.4.1.2 V) IWPAttrInterface.GetUnderlineColorNr

Applies to
IWPAttrInterface

Declaration
```
function GetUnderlineColorNr(var ColorNr: Integer): Boolean;
```

Description
This method reads the color used for underlines as index value.

4.4.1.2 W) IWPAttrInterface.GetUnderlineMode

Applies to
IWPAttrInterface

Declaration
```
function GetUnderlineMode(var Mode: Integer): Boolean;
```

Description
This method reads the mode used for underlines. This are the available values.

4.4.1.2 X) IWPAttrInterface.GetWPCSS

Get Definition as string.

Applies to
IWPAttrInterface

Declaration
```
function GetWPCSS: WideString;
```

Description
This function creates a string with the values of all properties which are defined (which are not "default").
This method is very useful to quickly save the current attributes to be applied later using SetWPCSS.

Category
TextDynamic CSS strings

4.4.1.2 Y) IWPAttrInterface.NrToColor

Applies to
IWPAttrInterface

Declaration
```
string NrToColor(int NrToColor);
```

Description
The engine internally saves colors as index values. This method converts an index into a RGB value.

The method can be used to convert the IDs the AttrGet function will retrieve. GetColor and GetBGColor will already return a color value.

4.4.1.2 Z) IWPAttrInterface.SetBGColor

Applies to
IWPAttrInterface

Declaration
```
procedure SetBGColor(RGB: TColor);
```

Description
This method is used to set the background color for characters, also known as highlighting. The parameter is a RGB value.

In .NET applications you can use the utility function **ToRGB** to convert a *Color* member into a RGB value. Example:

```
WPDLLInt1.ToRGB(Color.Red)
.
```

4.4.1.2 AA) IWPAttrInterface.SetBGColorNr

Applies to
IWPAttrInterface

Declaration
```
void SetBGColorNr(int ColorNr);
```

Description
This method is used to set the background color for characters, also known as highlighting. The parameter is the internal index value used by the RTF engine.

The methods ColorToNr and NrToColor are used to convert between index values and RGB values.

4.4.1.2 AB) IWPAttrInterface.SetCharStyleSheetName

Applies to
IWPAttrInterface

Declaration
```
procedure SetCharStyleSheetName(const Name: WideString);
```

Description
This method is used to assign a style to characters. The parameter is the name of a paragraph style. Currently SetCharStyleSheetName is not used, it is reserved for future use.

4.4.1.2 AC) IWPAttrInterface.SetCharWidth

Applies to
IWPAttrInterface

Declaration
```
procedure SetCharWidth(CharWidthPC: Integer);
```

Description
With this method the spacing between characters can be controlled. Values >$8000 are interpreted as negative values. Use the parametr -1 to remove the character spacing.

4.4.1.2 AD) IWPAttrInterface.SetColor / ForeColor

Applies to
IWPAttrInterface

Declaration
```
procedure SetColor(Color: TColor);
```

Description
This method is used to set the font color for characters. The parameter is a RGB value.

In .NET applications you can use the utility function **ToRGB** to convert a *Color* member into a RGB value. Example:

```
WPDLLInt1.ToRGB(Color.Red)
.
```

4.4.1.2 AE) IWPAttrInterface.SetColorNr

Applies to
IWPAttrInterface

Declaration
```
void SetColorNr(int ColorNr);
```

Description
This method is used to set the font color. The parameter is the internal index value used by the RTF engine.

The methods ColorToNr and NrToColor are used to convert between index values and RGB values.

4.4.1.2 AF) IWPAttrInterface.SetFontCharset

Applies to
IWPAttrInterface

Declaration
```
procedure SetFontCharset(Charset: Integer);
```

Description
This methods sets the charset for the text. TextDynamic internally works with unicode values, however Charsets are important for the conversion of ANSI strings to unicode, for the displays of symbol fonts and for PDF export. Usually when text is typed the charset will be retrieved from the OS.

4.4.1.2 AG) IWPAttrInterface.SetFontface

Applies to
IWPAttrInterface

Declaration
```
procedure SetFontface(const FontName: WideString);
```

Description
This method is used to set the name of the font used by the text.

4.4.1.2 AH) IWPAttrInterface.SetFontSize

Applies to
IWPAttrInterface

Declaration
```
procedure SetFontSize(Size: Single);
```

Description
Use this method to set the font height in point. The parameter is a floating point value.

4.4.1.2 AI) IWPAttrInterface.SetStyles

Applies to
IWPAttrInterface

Declaration
```
procedure SetStyles(Value: WrtStyle);
```

Description
This method sets all character styles ate once. Also see IncludeStyles and ExcludeStyles and ToogleStyle.

Parameters	
Styles	**1 :** Bold text. (C# wrapper defines enum element WPSTY.BOLD) **2 :** Italic text. (C# wrapper defines enum element WPSTY.ITALIC) **4 :** Underlined text. (C# wrapper defines enum element WPSTY.UNDERLINE) **8 :** Strikeout text. (C# wrapper defines enum element WPSTY.STRIKEOUT) **16 :** Text in super-script (C# wrapper defines enum element WPSTY.SUPERSCRIPT) **32 :** Text in sub-script (C# wrapper defines enum element WPSTY.SUBSCRIPT) **64 :** Hidden text, (C# wrapper: WPYST. HIDDEN) **128 :** Uppercase text. (C# wrapper: WPYST. UPPERCASE) **256 :** reserved. **512 :** Lowercase text. (C# wrapper: WPYST. LOWERCASE) **1024 :** Text which should be excluded from spellcheck (WPSTY.NOPROOF) **2048 :** Double strikeout (WPSTY. DBLSTRIKEOUT) **4096 :** reserved. **8192 :** protected text (WPSTY.PROTECTED)

Category
Character Styles

4.4.1.2 AJ) IWPAttrInterface.SetTextLanguage

Applies to
IWPAttrInterface

Declaration
```
procedure SetTextLanguage(Mode: Integer);
```

Description
This method is used to set the language for the text. This method is currently not used.

4.4.1.2 AK) IWPAttrInterface.SetUnderlineColor

Applies to
IWPAttrInterface

Declaration
```
procedure SetUnderlineColor(Color: TColor);
```

Description
This method sets the color for the underline as RGB value. See example "Underline Modes".

4.4.1.2 AL) IWPAttrInterface.SetUnderlineColorNr

Applies to
IWPAttrInterface

Declaration
```
procedure SetUnderlineColorNr(ColorNr: Integer);
```

Description
This method sets the color for the underline as color index value.

4.4.1.2 AM) IWPAttrInterface.SetUnderlineMode

Applies to
IWPAttrInterface

Declaration
```
procedure SetUnderlineMode(Mode: Integer);
```

Description
This method sets the mode used for underlines. See GetUnderlineModes.

4.4.1.2 AN) IWPAttrInterface.SetWPCSS

Applies to
IWPAttrInterface

Declaration
```
procedure SetWPCSS(const wpcss: WideString);
```

Description
Set the attributes saved to a string using GetWPCSS.

Category
TextDynamic CSS strings

4.4.1.2 AO) IWPAttrInterface.ToggleStyle

Applies to
IWPAttrInterface

Declaration
```
procedure ToggleStyle(Element: Integer);
```

Description
This method changes a certain character attribute flag from unset to set and vice versa. You can use it in the event OnKeyPress to implement a hotkey to enable/disable a certain mode, such as 'bold text'.
VB6:

```
Private Sub WPDLLInt1_OnKeyPress(ByVal Editor As Long, Key As Byte)
  Dim Memo As IWPMemo
  If Editor = 2 Then Set Memo = WPDLLInt1.Memo2 Else: Set Memo = WPDLLInt1.Memo
  If Key = 2 Then ' Ctrl B
      If Memo.TextCursor.IsSelected Then
         Memo.CurrSelAttr.ToggleStyle (0)
      Else
         Memo.CurrAttr.ToggleStyle (0) ' set bold!
      End If
      Key = 0
  End If
End Sub
```

C#:
```
private void WPDLLInt1_KeyPress(object sender, System.Windows.Forms.KeyPressEventArgs e)
{
     IWPMemo Memo;
     if (((WPDynamic.wpKeyPressEventArgs)e).Editor==2 )
    Memo = WPDLLInt1.Memo2;
     else Memo = WPDLLInt1.Memo;
     if (e.KeyChar==(Char)2) // Ctrl+B to toggle "Bold"
     { if (Memo.TextCursor.IsSelected)
    Memo.CurrSelAttr.ToggleStyle((int)WPWRT.BOLD);
    else Memo.CurrSelAttr.ToggleStyle((int)WPWRT.BOLD);
     }
}
```

VB6 (see intro):

```
        Case "Fett"
            ActiveForm.rtfText.CurrAttr.ToggleStyle 0
        Case "Kursiv"
            ActiveForm.rtfText.CurrAttr.ToggleStyle 1
        Case "Unterstrichen"
            ActiveForm.rtfText.CurrAttr.ToggleStyle 2
```

Parameters	
Element	**0 :** Bold text. (C# wrapper defines enum element WPWRT.BOLD) **1 :** Italic text. (C# wrapper defines enum element WPWRT.ITALIC)

	2 : Underlined text. (C# wrapper defines enum element WPWRT.UNDERLINE) **3 :** Strikeout text. (C# wrapper defines enum element WPWRT.STRIKEOUT) **4 :** Text in super-script (C# wrapper defines enum element WPWRT.SUPERSCRIPT) **5 :** Text in sub-script (C# wrapper defines enum element WPWRT.SUBSCRIPT) **6 :** Hidden text, (C# wrapper: WPWRT.HIDDEN) **7 :** Uppercase text. (C# wrapper: WPWRT.UPPERCASE) **8 :** reserved. **9 :** Lowercase text. (C# wrapper: WPWRT.LOWERCASE) **10 :** Text which should be excluded from spellcheck (WPWRT.NOPROOF) **11 :** Double strikeout (WPWRT.DBLSTRIKEOUT) **12 :** reserved. **13 :** protected text (WPWRT.PROTECTED)

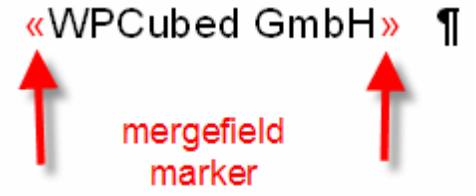
Category
Character Styles

4.4.2 IWPCharacterAttr

Attributes for links and fields

Description
This interface is used to manipulate the appearance of certain "special" text. Such special text are hyperlinks, protected text, field objects and the markers which embedd merged text (=insertpoints).

To change the text which is displayed by start and end markers You can use the method SetCodeText. Use first parameter =1 to change start marker, 2 to change end marker.
Create a merge field:
wpdllInt1.TextCursor.InputField("NAME","WPCubed GmbH",false);



Now change the CodeText to display ( ) signs.
wpdllInt1.SpecialTextAttr(SpecialTextSel.wpInsertpoints).SetCodeText(**1**,"'(");
wpdllInt1.SpecialTextAttr(SpecialTextSel.wpInsertpoints).SetCodeText(**2**,"')");
(WPCubed GmbH)
It is also possible to display the name in one of the markers using the string "%N(":
NAME(WPCubed GmbH) (%N displays the name, %S displays the Source property of the text object, %Y displays the name of the style used by this object, %P displays the Params property.)

Available ids for function SpecialTextAttr:

```
0 : wpHiddenText - hidden text
1 : wpFootnote - footnote symbols
2 : wpInsertpoints - the star and end markers of merge fields
3 : wpHyperlink - the start and end marker of hyperlinks
4 : wpSPANStyle - the start and end marker of inline SPAN styles
5 : wpAutomaticText - merged text within fields
6 : wpProtectedText - the text which has the protected property
7 : wpBookmarkedText - the text within bookmark objects
8 : wpInsertedText - not used
9 : wpDeletedText - not used
10: wpWordHighlight - highlighted words
11: wpFieldTextObjects - single objects, such as page numbers.
```

Properties	Properties	Method
BackgroundColor	Italic	SetCodeText
Bold	StrikeOut	
CodeTextColor	SubScript	
DoubleUnderline	SuperScript	
Hidden	TextColor	
HotEffect	Underline	
HotTextColor	UnderlineColor	

The properties Bold, Italic, StrikeOut, SubScript, SuperScript and Underline use the type **ThreeState**. It is an enum with this values:

tsIgnore = 0: do not use the property
tsTrue = 1: Swith mode always on
tsFalse = 2: Switch this mode always off

4.4.2.1 Properties

4.4.2.1 A) BackgroundColor

Applies to
IWPCharacterAttr

Declaration
```
int BackgroundColor;
```

Description
The background color as RGB value, can be used for merged text, bookmarked text and hyperlinks.

4.4.2.1 B) Bold

Applies to
IWPCharacterAttr

Declaration
```
ThreeState Bold;
```

Description
Use a bold font, can be used for merged text, bookmarked text and hyperlinks.

4.4.2.1 C) CodeTextColor

Applies to
IWPCharacterAttr

Declaration
```
property CodeTextColor: Integer read Get_CodeTextColor write Set_CodeTextColor;
```

Description
The color used to display the object markers used by merge fields and bookmarks.

4.4.2.1 D) DoubleUnderline

Applies to
IWPCharacterAttr

Declaration
```
property DoubleUnderline: WordBool read Get_DoubleUnderline write
Set_DoubleUnderline;
```

Description
Use double underlining, can be used for merged text, bookmarked text and hyperlinks.

4.4.2.1 E) Hidden

Applies to
IWPCharacterAttr

Declaration
```
bool Hidden;
```

Description
Hide the code objects

4.4.2.1 F) HotEffect

Applies to
IWPCharacterAttr

Declaration
```
property HotEffect: Integer read Get_HotEffect write Set_HotEffect;
```

Description
When the mouse is moved over the text it is automatically highlighted. (hover effect)

4.4.2.1 G) HotTextColor

Applies to
IWPCharacterAttr

Declaration
```
property HotTextColor: Integer read Get_HotTextColor write Set_HotTextColor;
```

Description
When the mouse is moved over the text it is automatically highlighted. (hover effect)

4.4.2.1 H) Italic

Applies to
IWPCharacterAttr

Declaration
```
ThreeState Italic;
```

Description
Use an italic font, can be used for merged text, bookmarked text and hyperlinks.

4.4.2.1 I) StrikeOut

Applies to
IWPCharacterAttr

Declaration
```
ThreeState StrikeOut;
```

Description
```
Use tsTrue (value=1) to strike out the text.
```

4.4.2.1 J) SubScript

Applies to
IWPCharacterAttr

Declaration
```
ThreeState SubScript;
```

Description
Use a subscript font.

4.4.2.1 K) SuperScript

Applies to
IWPCharacterAttr

Declaration
```
ThreeState SuperScript;
```

Description
Use a superscript font.

4.4.2.1 L) TextColor

Applies to
IWPCharacterAttr

Declaration
```
ThreeState TextColor;
```

Description
The color for the embedded text. This does not change the color for the objects.

4.4.2.1 M) Underline

Applies to
IWPCharacterAttr

Declaration
```
ThreeState Underline;
```

Description
Use underlining.

4.4.2.1 N) UnderlineColor

Applies to
IWPCharacterAttr

Declaration
```
int UnderlineColor;
```

Description
Change the color for the underline.

4.4.2.2 Methods

4.4.2.2 A) IWPCharacterAttr.SetCodeText

Applies to
IWPCharacterAttr

Declaration
```
procedure SetCodeText(Select: Integer; const Text: WideString);
```

Description
To change the text which is displayed by start and end markers You can use the method SetCodeText. Use first parameter =1 to change start marker, 2 to change end marker. The string may contain placeholders: "%N" displays the name, "%S" displays the Source property of the text object, "%Y" displays the name of the style used by this object, "%P" displays the Params property.

SetCodeText Example

```
wpdllInt1.TextCursor.InputField("NAME","WPCubed GmbH",false);   wpdllInt1.SpecialTextAttr
```

4.5 Text Elements (paragraphs, objects, styles)

Using IWPParInterface you get access to the current paragraph. You can change the text or change the attributes.

IWPParInterface is also used to modify paragraph styles. Here you cannot add text but You can change the attributes.

This three basic function IWPAttrInterface.AttrDel, IWPAttrInterface.AttrGet, IWPAttrInterface.AttrSet will modify the character attributes for IDs <=16, for IDs >16 the current paragraph / selected paragraphs will be modified. You can use Memo.TextAttr.AttrSET((int)WPAT.IndentLeft, 360) to change the indent of the current paragraph or all selected paragraphs, in case text was selected.

IWPDataBlock provides low level access to the paragraph tree. You can use it to append paragraphs and to delete header or footer.

IWPTextObj provides access to objects such as field markers and hyperlinks, but also images.

IWPNumberStyle can be used to create and modify numbering styles.

4.5.1 IWPParInterface

Low level paragraph text and attribute access

Description
The IWPParInterface makes it possible to manipulate text styles and paragraphs.

If you use it to add text (instead of IWPTextCursor functions) the new text can be creation without the need to move the cursor position. This can be useful to create text "on the fly" in headers or footers (see IWPDataBlock.AppendParagraph).

If You use Memo.CurrPar usually only the current paragraph, the paragraph the cursor is in, will be modified. **But many methods and properties can also modify all selected paragraphs**.

To do so, You first need to call SetProp(1,1) and when done SetProp(1,0). We highlighted the methods with this ability with this symbol:
Use SetProp(1,1) to work with selected text.

This codes appends text to the paragraph:

```
IWPAttrInterface atr = wpdllInt1.AttrHelper;
IWPParInterface par = memo.CurrPar;
// Append normal text
atr.Clear();
par.AppendText("Normal ", atr.CharAttrIndex);
// bold text
atr.IncludeStyles(1);
par.AppendText("and bold", atr.CharAttrIndex);
```

IWPParInterface is used to change the properties of a paragraph:

```
IWPParInterface par= Memo.CurrPar;
par.IndentLeft = (int)(3/2.54*1440); // Indent left = 3 cm
par.IndentRight = (int)(1.5*1440); // Indent right = 1.5 inch
```

In a **data bound editor** you need to set the modified flag prior to the change of the text. You can use
the method TextCursor.CheckState(10) to do it. This will trigger the PropChange

event.

IWPParInterface is also used to change a style element.

```
WPDynamic.IWPParInterface style;
int i;
// create a new style
wpdllInt1.Memo.SelectStyle("New_Style");
// get interface to change the style
style = wpdllInt1.Memo.CurrStyle;
// set properties in stlye
style.Alignment = 2;
// select the style for the current paragraph
wpdllInt1.CurrPar.StyleName = "New_Style";
```

If you need to change the NextStyle property of a paragraph style please use ConvertTextToValue:

```
// Interface to modify the properties of the style
WPDynamic.IWPParInterface style = wpdllInt1.Memo.CurrStyle;
int i;

// Create style 1
wpdllInt1.Memo.SelectStyle("Style1");
style.Alignment = 1;
// Set NextStyle property
i = style.ConvertTextToValue("Style2");
style.ParASet((int)WPAT.STYLE_NEXT_NAME, i);

// Create style 2
wpdllInt1.Memo.SelectStyle("Style2");
style.Alignment = 2;
```

Tip: A table row can be duplicated using the CurrPar interface. Here we first let the CurrPar interface modify the current row and then call Duplicate():

```
Cursor.CPTableRowNr = 1; // goto row 1
Memo.CurrPar.SetPtr( Cursor.CPRowPtr );
while(rowcount-->0) Memo.CurrPar.Duplicate();
```

Hint:
It is possible to assign a **name** to a paragraph using ParStrCommand(4,0,name)

The cursor can be moved to a name paragraph using TextCursor.

Please note that the interface referenced by CurrPar can change for different paragraphs.

This demo will create some random text with increasing font sizes:

```
Memo.CurrPar.AppendNext();
for (int i = 0; i < 30; i++)
{
      Memo.TextCommand(31, 1);
      Memo.CurrPar.ParASet((int)WPAT.CharFontSize, (i + 1) * 200);
```

```
            Memo.CurrPar.AppendNext();
        }
```

Overview:

Properties:	Methods	Methods
Alignment	AppendChild	ParAAddBits
AlignmentVert	AppendNext	ParAClear
Borders	AppendText	ParADel
CellCommand	CharAttr	ParADelBits
CellName	CharObj	ParAGet
CharCount	ClearCharAttr	ParAInc
IndentFirst	DeleteChar	ParASet (also see Convert utility for font name and color values)
IndentLeft	DeleteParagraph	ParCommand
IndentRight	DeleteParEnd	ParStrCommand
IsColMerge	Duplicate	ReplaceCharAttr
IsFooterRow	GetAllText	ReplaceText
IsHeaderRow	GetChar	SaveToString
IsHidden	GetCharAttr	SetChar
IsNewPage	GetParType	SetCharAttr
IsProtected	GetProp	SetParType
IsRowMerge	GetPtr	SetProp
LineHeight	GetPtrChild	SetPtr
NumberLevel	GetPtrNext	SetText
NumberMode	GetPtrParent	TabAdd
ParColor	GetPtrPrev	TabClear
ParCSS	GetSubText	TabDelete
ParShading	GetText	Low Level Move Methods (Select..)
ParWPCSS	HasText	
SpaceAfter	InsertNewObject	
SpaceBefore	InsertText	
SpaceBetween	LoadFromFile	
StyleName	LoadFromString	
TOCOutlineLevel		
WidthTW		

4.5.1.1 Properties

4.5.1.1 A) Alignment

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
int Alignment;
```

Description
The alignment (WPAT_Alignment) used by this paragraph or style:
0 : Left aligned
1 : Centered
2 : Right aligned
3 : Justified Text

4.5.1.1 B) AlignmentVert

Applies to
IWPParInterface
Use SetProp(1,1) to work with selected text.

Declaration
```
int AlignmentVert;
```

Description
Vertical alignment (WPAT_VertAlignment) - only possible in cells:
0 : top aligned
1 : centered vertically
2 : bottom aligned

4.5.1.1 C) Borders

Applies to
IWPParInterface
Use SetProp(1,1) to work with selected text.

Declaration
```
int Borders;
```

Description
Borders (WPAT_BorderFlags) used by this paragraph (styles do not yet use borders).
Bits:
1 : Left border
2 : Top border
4 : Right border
8 : Bottom border
15: all 4 borders.

4.5.1.1 D) CellCommand

Applies to
IWPParInterface
Use SetProp(1,1) to work with selected text.

Declaration
```
string CellCommand;
```

Description
The formula used by this cell. Used for calculation in tables.

4.5.1.1 E) CellName

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
string CellName;
```

Description
The cell name used by this cell. Used for calculation in tables.

To assign a name to the paragraph which can be searched for, use ParStrCommand(4,name)

4.5.1.1 F) CharCount

Applies to
IWPParInterface

Declaration
```
int CharCount;
```

Description
The count of characters in a paragraph.

4.5.1.1 G) IndentFirst

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
int IndentFirst;
```

Description
The first indent (WPAT_IndentFirst) as twips value.

4.5.1.1 H) IndentLeft

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
int IndentLeft;
```

Description
The left indent (WPAT_IndentLeft) as twips value. Use IndentFirst=-IndentLeft for hanging indents.

4.5.1.1 I) IndentRight

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
int IndentRight;
```

Description
The right indent (WPAT_IndentRight) as twips value.

4.5.1.1 J) IsColMerge

Applies to
IWPParInterface

Declaration
```
bool IsColMerge;
```

Description
When set, this paragraph (must be a cell) is horizontally merged with the previous cell.

Please also see InputRowStart - there we added an VB.NET example which creates a table using merged cells and paragraph styles.

4.5.1.1 K) IsFooterRow

Applies to
IWPParInterface

Declaration
```
bool IsFooterRow;
```

Description
When set this row paragraph (or the parent row if it is a cell) is a footer row. Footer rows are repeated at the bottom of the page.

Category
Table Support

4.5.1.1 L) IsHeaderRow

Applies to
IWPParInterface

Declaration
```
bool IsHeaderRow;
```

Description
When set this row paragraph (or the parent row if it is a cell) is a header row. Header rows are repeated at the top of the page.

Category
Table Support

4.5.1.1 M) IsHidden

Applies to
IWPParInterface

Declaration
```
bool IsHidden;
```

Description
If set this paragraph (must not be a cell) is hidden.

4.5.1.1 N) IsNewPage

Applies to
IWPParInterface

Declaration
```
bool IsNewPage;
```

Description
A new page starts before this paragraph. In tables hard pagebreaks are only allowed before rows, soft page breaks are allowed within cells.

4.5.1.1 O) IsProtected

Applies to
IWPParInterface

Declaration
```
bool IsProtected;
```

Description
If set this paragraph is protected.

4.5.1.1 P) IsRowMerge

Applies to
IWPParInterface

Declaration
```
bool IsRowMerge;
```

Description
When set, this paragraph (must be a cell) is vertically merged with the previous cell.

Also see property IsColMerge.

Category
Table Support

4.5.1.1 Q) LineHeight

Applies to
IWPParInterface
Use SetProp(1,1) to work with selected text.

Declaration
```
int LineHeight;
```

Description
The line height in percent (WPAT_LineHeight) defined for this paragraph or style.

4.5.1.1 R) NumberLevel

Applies to
IWPParInterface
Use SetProp(1,1) to work with selected text.

Declaration
```
int NumberLevel;
```

Description
The outline level of this style or paragraph. Possible values are between 0 and 9. 0 disables the outline mode.
The number level can also be modified using property ID WPAT_NumberLevel.

Example:
```
   IWPTextCursor Cursor = rtF2PDF1.Memo.TextCursor;
   IWPParInterface Par =  rtF2PDF1.Memo.CurrPar;

   IWPNumberStyle NStyle = rtF2PDF1.Memo.GetNumberStyle(0, 0, 1);
   NStyle.ASet((int)WPAT.NumberMODE,1); // Arabic
   rtF2PDF1.ReleaseInt(NStyle);

   Cursor.InputText("This is the first outline");
   Cursor.InputParagraph(0,"");
   for (int i = 0; i < 10; i++)
   {
               Cursor.InputText("List Item " + i.ToString());
               Par.NumberLevel=1;
               Cursor.InputParagraph(0,"");
   }
   Par.NumberMode = 0;

   Cursor.InputText("This is the second outline");
   Cursor.InputParagraph(0,"");

   for (int i = 0; i < 10; i++)
   {
               Cursor.InputText("List Item " + i.ToString());
               Par.NumberLevel=1;
               // The following code makes sure the numbering is restarted
               if (i==0)
               Par.ParASet((int)WPAT.NumberStart, 1);
               Cursor.InputParagraph(0,"");
   }
```

```
Par.NumberMode = 0;
```

4.5.1.1 S) NumberMode

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
int NumberMode;
```

Description
The numbering mode for this paragraph or style. If you assign a value NumberLevel will be cleared.
Possible values are:
0 : no numbering
1 : bullets
2 : circles (not used)
3 : arabic numbering 1,2,3
4 : captital roman
5 : roman
6 : capital latin
7 : latin
Note, this property is not stored as property with WPAT code "WPAT_NumberMode". WPAT_NumberMode is not used by paragraph styles or paragraphs, only by numberstyles. To get a numberstyle the use the ID stored as property WPAT_NumberStyle with GetNumberStyle.

Example - "Simple numbering":
```
IWPTextCursor Cursor = rtF2PDF1.Memo.TextCursor;
IWPParInterface Par =  rtF2PDF1.Memo.CurrPar;
Cursor.InputText("This is not numbered text");
Cursor.InputParagraph(0,"");
for (int i = 0; i < 10; i++)
{
            Cursor.InputText("List Item " + i.ToString());
            Par.NumberMode = 3;
            Cursor.InputParagraph(0,"");
}
Par.NumberMode = 0;
Cursor.InputText("This is not numbered text");
```

Example - Multi Level Numbering:
```
IWPTextCursor Cursor = rtF2PDF1.Memo.TextCursor;
IWPParInterface Par =  rtF2PDF1.Memo.CurrPar;

IWPNumberStyle NStyle = rtF2PDF1.Memo.GetNumberStyle(0, 0, 1);
NStyle.ASet((int)WPAT.NumberMODE,1); // Arabic
rtF2PDF1.ReleaseInt(NStyle);

Cursor.InputText("This is not numbered text");
Cursor.InputParagraph(0,"");
for (int i = 0; i < 10; i++)
{
            Cursor.InputText("List Item " + i.ToString());
            Par.NumberLevel=1;
            Cursor.InputParagraph(0,"");
```

```
    }
    Par.NumberMode = 0;
    Cursor.InputText("This is not numbered text");
```

4.5.1.1 T) ParColor

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
int ParColor;
```

Description
The paragraph shading color as RGB value.

Example:

This VB.NET code creates a paragraph style with blue shading. It uses the ColorToRGB utility.
```
Memo.SelectStyle("DataRow")
Memo.CurrStyle.ParColor = WpdllInt1.ColorToRGB(Color.Blue)
Memo.CurrStyle.ParShading = 20
```

4.5.1.1 U) ParCSS

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
property ParCSS: WideString read Get_ParCSS write Set_ParCSS;
```

Description
Sets and retrieves paragraph or style attributes in standard CSS Level 2 syntax.

Category
TextDynamic CSS strings

4.5.1.1 V) ParShading

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
int ParShading;
```

Description
The paragraph shading as percent value. Also see property ParColor.

4.5.1.1 W) ParWPCSS

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
string ParWPCSS;
```

Description
Read and write the paragraph attributes in WPCSS format.

Category
TextDynamic CSS strings

4.5.1.1 X) SpaceAfter

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
int SpaceAfter;
```

Description
The space after the paragraph as twips value.

4.5.1.1 Y) SpaceBefore

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
int SpaceBefore;
```

Description
The space before the paragraph as twips value.

4.5.1.1 Z) SpaceBetween

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
int SpaceBetween;
```

Description
The line height as twips value. If positive it will be used as minimum height, if negative it will be used as absolute height. This property overrules LineHeight.

4.5.1.1 AA) StyleName

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration

```
string StyleName;
```

Description
The name of the paragraph style used by this paragraph. For paragraph styles, this is the name of the base style.

You can add a style using Memo.SelectStyle(name)

Example - inside the OnEnumParOrStyle event create and apply set a certain style

```
private void wpdllInt1_OnEnumParOrStyle(object Sender,
  bool IsControlPar, int StartPos, int Count,
  WPDynamic.IWPParInterface ParText,
  WPDynamic.IWPAttrInterface ParAttr,
  int EventParam, ref bool Abort)
{
     wpdllInt1.Memo.SelectStyle( new_style_name );
     wpdllInt1.Memo.CurrStyleAttr.SetFontSize( size_for_style );
     ParText.ClearCharAttr();
     ParText.StyleName = s;
     Abort = false;
}
```

Please note that the attributes in a style are only used, if they are not overridden by the paragraph or character attributes. Here we use ClearCharAttr to make sure, the characters do not have their own attributes.

For a more selective control apply the style using ParStrCommand(9, x, name). If bit 1 is set in X, the character attributes, which are used by the selected style, are cleared, if bit 2 is set, the paragraph attributes are cleared.

Also see: "Paragraph Styles".

You need to call Memo.ReformatAll(true, true) after you have applied styles!

Category
Paragraphstyle Support

4.5.1.1 AB) TOCOutlineLevel

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
int TOCOutlineLevel;
```

Description
Outline level used in table of contents. When creationg a TOC all paragraphs which use a value>0 will be collected. (Internally for TOCs the attribute with the ID WPAT_ParIsOutline is used)

4.5.1.1 AC) WidthTW

Applies to
IWPParInterface

Declaration
```
int WidthTW;
```

Description
The current formatted width of this paragraph. This value can be used to measure cells, it is readonly.

Note: There are some commands available to read and set the **current row height** using the method ParCommand.

4.5.1.2 Methods to manage attributes

4.5.1.2 A) IWPParInterface.ParASet

Applies to
IWPParInterface

Declaration
```
procedure ParASet(WPAT_Code: Integer; Value: Integer);
```

Description
Sets an attribute.

The property is selected by the WPAT code.

Note: Internally the properties are stored in an array, not in specific variables. Only if a property is defined, an entry is added to this array. This does not only save resources, but it allows all properties to have an undefined state. This feature is important for a clean support of paragraph styles.

Also see Convert Utility for Fontface & Color Values.

4.5.1.2 B) IWPParInterface.ParAInc

Applies to
IWPParInterface

Declaration
```
procedure ParAInc(WPAT_Code: Integer; Offset: Integer; MinValue: Integer);
```

Description
Increments attribute. The last parameter is the minimum allowed value.

4.5.1.2 C) IWPParInterface.ParAGet

Applies to
IWPParInterface

Declaration
```
function ParAGet(WPAT_Code: Integer; var Value: Integer): WordBool;
```

Description
Retrieve value of attribute. Retruns true if attribute is defined, otherwise false.

Also see "convert utility functions".

4.5.1.2 D) IWPParInterface.ParADelBits

Applies to
IWPParInterface
Use SetProp(1,1) to work with selected text.

Declaration
```
procedure ParADelBits(WPAT_Code: Integer; Bits: Integer);
```

Description
Performs "AND NOT value" operation with specified value to delete bits.

4.5.1.2 E) IWPParInterface.ParADel

Applies to
IWPParInterface
Use SetProp(1,1) to work with selected text.

Declaration
```
procedure ParADel(WPAT_Code: Integer);
```

Description
Deletes a certain attribute. This makes this attribute undefined.

Also see Convert utility for Fontface & Color Values.

4.5.1.2 F) IWPParInterface.ParAClear

Applies to
IWPParInterface
Use SetProp(1,1) to work with selected text.

Declaration
```
procedure ParAClear(Mode: Integer);
```

Description
Deletes all attributes stored in this paragraph or style.

If bit 1 is set in parameter mode the character attribute in a paragraph are also cleared. (In contrast to ClearCharAttr this does not change children paragraphs!)

4.5.1.2 G) IWPParInterface.ParAAddBits

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
procedure ParAAddBits(WPAT_Code: Integer; Bits: Integer);
```

Description
Perform "OR value" operation with specified attribute value to add bits.

You can use this method to set the character styles. Here you need to set the mask and the on/off flag:

```
Memo.CurrPar.ParAAddBits((int)WPAT.CharStyleON, 1+4);
Memo.CurrPar.ParAAddBits((int)WPAT.CharStyleMask, 1+4);
```

The following bits can be used:

bold : 1;
italic : 2;
underlined : 4;
strikeout : 8;
super script: 16;
sub script: 32;
hidden: 64
uppercase: 128;
smallcaps: 256;
lowercase: 512;
no proof (disable spellcheck) : 1024;
double strikeout: 2048;
reserved: 4096;
protected text: 8192;

Please note that paragraph style attributes are only used by text which does not define the same attribute itself.

You can use ClearCharAttr to normalize a complete paragraph.

4.5.1.2 H) IWPParInterface.SetCharAttr

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
procedure SetCharAttr(Index: Integer; CharAttr: Integer);
```

Description
This methods sets the character attribute index of one character in the paragraph.
This example use EnumParagraphs to set one character attribute (Courier New, 10.5 pt, Green) to the complete text.

```
IWPAttrInterface help = wpdllInt1.AttrHelper;
```

```
help.Clear();
help.SetFontface("Courier New");
help.SetColor(wpdllInt1.ToRGB(Color.Green));
help.SetFontSize(10.5F);
// Trigger OnEnumParOrStyle
wpdllInt1.CurrMemo.EnumParagraphs(true,help.CharAttrIndex);
wpdllInt1.CurrMemo.ReformatAll(true,true);
```

This event handler executed for all paragraphs:

```
private void wpdllInt1_OnEnumParOrStyle(
 object Sender,
 bool IsControlPar,
 int StartPos,
 int Count,
 WPDynamic.IWPParInterface ParText,
 WPDynamic.IWPAttrInterface ParAttr,
 int EventParam, //= int parameter passed to EnumParagraphs
 ref bool Abort)
{
 for (int i = 0; i < ParText.CharCount; i++)
  ParText.SetCharAttr(i, EventParam);
}
```

Note: The example above is intended to only demonstrate the technique, to set the character attribute of the complete text we recommend to used code like this:

```
IWPAttrInterface SelAttr = wpdllInt1.CurrSelAttr;
wpdllInt1.TextCursor.SelectAll();
SelAttr.SetFontface("Courier New");
SelAttr.SetColor(wpdllInt1.ToRGB(Color.Green));
SelAttr.SetFontSize(10.5F);
wpdllInt1.TextCursor.HideSelection();
```

Category
Character Attributes

4.5.1.2 I) IWPParInterface.ReplaceCharAttr

Applies to
IWPParInterface

Declaration

```
procedure ReplaceCharAttr(PosInPar: Integer; Len: Integer; NewCharAttrIndex:
Integer);
```

Description
Replaces certain CharAttr index values with others.

If you need to replace certain properties, i.e. make all "bold" text "italic", You can use the command #7 and #8 of ParCommand().

Category
Character Attributes

4.5.1.2 J) IWPParInterface.GetCharStyleAt

This function can be used to read character attributes of text within the paragraph. We added this method as easier alternative to the use of CharAttr.

The method GetCharFontAt reads the font styes at a given position.

The result is a bitfield:
1 : Bold text. (C# wrapper defines enum element WPSTY.BOLD)
2 : Italic text. (C# wrapper defines enum element WPSTY.ITALIC)
4 : Underlined text. (C# wrapper defines enum element WPSTY.UNDERLINE)
8 : Strikeout text. (C# wrapper defines enum element WPSTY.STRIKEOUT)
16 : Text in super-script (C# wrapper defines enum element WPSTY.SUPERSCRIPT)
32 : Text in sub-script (C# wrapper defines enum element WPSTY.SUBSCRIPT)
64 : Hidden text, (C# wrapper: WPYST.HIDDEN)
128 : Uppercase text. (C# wrapper: WPYST.UPPERCASE)
256 : reserved.
512 : Lowercase text. (C# wrapper: WPYST.LOWERCASE)
1024 : Text which should be excluded from spellcheck (WPSTY.NOPROOF)
2048 : Double strikeout (WPSTY.DBLSTRIKEOUT)
4096 : reserved.
8192 : protected text (WPSTY.PROTECTED)

> Note: Using Command #7 in ParCommand there is an optimized find routine available. It makes it possible to search a paragraph for all occurrences of a certain attribute.

4.5.1.2 K) IWPParInterface.GetCharSizeAt

This function can be used to read character attributes of text within the paragraph. We added this method as easier alternative to the use of CharAttr.

The method GetCharFontAt reads the font size at a given position. If the character does not define a font size, the default font size is returned.

> Note: Using Command #7 in ParCommand there is an optimized find routine available. It makes it possible to search a paragraph for all occurrences of a certain attribute.

4.5.1.2 L) IWPParInterface.GetCharFontAt

This function can be used to read character attributes of text within the paragraph. We added this method as easier alternative to the use of CharAttr.

The method GetCharFontAt reads the font name at a given position. If the character does not define a font name, the default font name is returned.

> Note: Using Command #7 in ParCommand there is an optimized find routine available. It makes it possible to search a paragraph for all occurrences of a certain attribute.

4.5.1.2 M) IWPParInterface.GetCharColorAt

This function can be used to read character attributes of text within the paragraph. We added this method as easier alternative to the use of CharAttr.

The method GetCharFontAt reads the text color as RGB value at a given position.

4.5.1.2 N) IWPParInterface.GetCharBGColorAt

This function can be used to read character attributes of text within the paragraph. We added this method as easier alternative to the use of CharAttr.

The method GetCharFontAt reads the text background color as RGB value at a given position.

4.5.1.2 O) IWPParInterface.GetCharAttr

Applies to
IWPParInterface

Declaration
```
function GetCharAttr(Index: Integer): Integer;
```

Description
This method ready the attribute index at a certain position in the paragraph.

Category
Character Attributes

4.5.1.2 P) IWPParInterface.ClearCharAttr

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
procedure ClearCharAttr;
```

Description
Sets the CharAttr ids of all the contained text and all children is set to to NEUTRAL. Also See ParAClear().

4.5.1.2 Q) IWPParInterface.CharAttr

The IWPAttrInterface to modify one character attribute in a paragraph.

Applies to
IWPParInterface

Declaration
```
function CharAttr(Index: Integer): IWPAttrInterface;
```

Description
This method returns an IWPAttrInterface reference which can be used to directly manipulate the attributes of one character in the paragraph. Please note that this interface is only valid until CharAttr is used the next time.

Example:
This code assigns the color red to all text which follows the comment signs "//". This example used the method EnumParagraphs which triggers the event OnEnumParOrStyle for all paragraphs in the text.

```
// Trigger event OnEnumParOrStyle
wpdllInt1.CurrMemo.EnumParagraphs(true,0);
// When done make sure the text is formatted and displayed
wpdllInt1.CurrMemo.ReformatAll(true,true);
```

This is the event handler for OnEnumParOrStyle:
```
private void wpdllInt1_OnEnumParOrStyle(
 object Sender,
 bool IsControlPar,
 int StartPos,
```

```
 int Count,
 WPDynamic.IWPParInterface ParText,
 WPDynamic.IWPAttrInterface ParAttr,
 int EventParam,
 ref bool Abort)
{
 for ( int i = 0; i < ParText.CharCount; i++)
  if (((char)ParText.GetChar(i)=='/')&&
      ((char)ParText.GetChar(i+1)=='/'))
  {
   while(i<ParText.CharCount)
   {
    ParText.CharAttr(i++).SetColor(
      wpdllInt1.ToRGB(Color.Red));
   }
  }
}
```

Note: If you need to apply a certain attribute to more text it is better to calculate an character attribute index (using AttrHelper) and then use SetCharAttr to apply it.

Also see:
IWPParInterface.GetCharFontAt
IWPParInterface.GetCharSizeAt
IWPParInterface.GetCharStyleAt
IWPParInterface.GetCharColorAt
IWPParInterface.GetCharBGColorAt

Category
Character Attributes

4.5.1.3 Methods to manage tab stops

4.5.1.3 A) IWPParInterface.TabstopDelete

Applies to
IWPParInterface

Declaration
```
void TabstopDelete(int Twips);
```

Description
Delete the tabstop at the specified position (twips value).

4.5.1.3 B) IWPParInterface.TabstopClear

Applies to
IWPParInterface

Declaration
```
procedure TabstopClear;
```

Description

Clears all tab stops in the paragraph or style.

To delete all tabs in the selected t3esxt use Memo.TextCommand(22)

4.5.1.3 C) IWPParInterface.TabAdd

Applies to
IWPParInterface
Use SetProp(1,1) to work with selected text.

Declaration
```
void TabAdd(int Twips, int Kind, int Fill, int ColorNr);
```

Description
Add (define) a tabstop in a paragraph.

To insert a paragraph at the current position use TextCursor.InputTabStop.

Also see: Tabstop Category

Parameters	
Twips	Value of this tabstop (in twips=1/1440 inch)
Kind	The tabstop kind: 0 : Left tab 1 : Right tab 2 : Center tab 3 : Decimal tab
Fill	The fill mode used for this tabstop 0: No Filling 1: Dots ..... 2: Dots in middle of line 3: Hyphens ------ 4: Underline _____ 5: Thick Hyphen **---------** 6: Equal Signs ======= 7: Arrow
ColorNr	0 or the color index. (Predefined index values are: 1=red, 2=green, 3=blue, 4=yellow, you can also use the convert method)

In addition there is a special operation mode:

TabAdd($FFFF, Fill, 0)

will add a right tabs top to the very end of the printable area.

Note: To add tabs in the selected text use Memo.TextCommand(ID,..):
ID: 21 - Adds a tabstop at position = paramA, mode + fill*256 = paramB. This method works with selections or the current paragraph.
ID: 22 - Deletes the tabstop at position paramA
ID: 23 - Clears all tabstops in the selected text or the current paragraph.

4.5.1.3 D) IWPParInterface.TabMove

Applies to

IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
procedure TabstopMove(OldValue, NewValue: Integer);

Moves a tabstop with a given twip value to a different position

4.5.1.3 E) IWPParInterface.TabGetNext

Applies to
IWPParInterface

Declaration

```
in TabGetNext(int AfterValue, int UntilValue);
```

Retrieves the tabstop value in twips within a given twip range

4.5.1.3 F) IWPParInterface.TabGet

Applies to
IWPParInterface

Declaration

This methods reads the properties of a tabstop with a given index.

Parameters:
Index: Integer;
Twips: Integer;
Align: Integer;
Color: Integer;
FillMode: Integer

Result=-1 in case the tabstop cannot be retrieved. Otherwise the value is 0.

You can use TabCount to retrieve the total count of tabs.

4.5.1.3 G) IWPParInterface.TabFind

Applies to
IWPParInterface

Declaration

int TabFind(int Value);

Finds a tabstop with a given twip value and returns the index.

4.5.1.3 H) IWPParInterface.TabCount

Applies to
IWPParInterface

Declaration

This method returns the total count of tabstops defined in this paragraph.

VB6 - Example, reads all tabstops one after the other

```
Dim c, i, pos, Align, Farbe, Leader

c = Ctl.CurrPar.TabCount
If c < 1 Then Exit Sub

For i = 0 To c - 1                          'get Tabs
    If Ctl.CurrPar.TabGet(i, pos, Align, Farbe, Leader) = 0 And pos > 0 Then
      ....
    End If
  End If
Next
```

4.5.1.4 Methods to manage text and images

4.5.1.4 A) IWPParInterface.SetText

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
procedure SetText(const NewText: WideString; CharAttr: Integer);
```

Description
Assigns the text of a paragraph

Parameters	
Text	New unicode text for the paragraph
CharAttr	CharAttr is the index of the character attribute record to be used for the inserted data. **Special values:** **0:** use the attribute of the preceding text or the current attribute in case the paragraph is empty. **-1:** use the attribute of the preceding text or the **paragraph** default attribute if at first position in paragraph. **-2:** use current writing mode (CurrAttr). **-3:** use the document default attribute **-4:** don't assign an attribute (CharAttr=0) **-5:** use the paragraphs default char attributes

4.5.1.4 B) IWPParInterface.AppendText

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
procedure AppendText(const NewText: WideString; CharAttr: Integer);
```

Description
This method appends text to the end of this paragraph. CharAttr can be usually passed as 0.

Parameters	
Text	Unicode string
CharAttr	CharAttr is the index of the character attribute record to be used for the inserted data. **Special values:** **0:**Use the attribute of the preceding text or the current attribute in case the paragraph is empty. **-1:**Use the attribute of the preceding text or the **paragraph** default attribute if at first position in paragraph. **-2:** use current writing mode (CurrAttr). **-3:** use the document default attribute **-4:** don't assign an attribute (CharAttr=0)

C# Example:

```
Memo.CurrAttr.Clear();
Memo.CurrAttr.SetFontface("Arial");
Memo.CurrAttr.SetFontSize(11);
Memo.CurrAttr.IncludeStyles(2); // Italic!
// append text using the current writing mode
Memo.CurrPar.AppendText("Some bold text", -2);
```

4.5.1.4 C) IWPParInterface.InsertText

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration

```
procedure InsertText(Index: Integer; const NewText: WideString; CharAttr:
Integer);
```

Description
Inserts text at a certain position in the paragraph

Parameters	
Index	Position to start insertion. Use a large value to append the text.
Text	Unicode string
CharAttr	CharAttr is the index of the character attribute record to be used for the inserted data. **Special values:** **0:**Use the attribute of the preceding text or the current attribute in case the paragraph is empty. **-1:**Use the attribute of the preceding text or the **paragraph** default attribute if at first position in paragraph. **-2:** use current writing mode (CurrAttr). **-3:** use the document default attribute **-4:** don't assign an attribute (CharAttr=0)

4.5.1.4 D) IWPParInterface.SetChar

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
procedure SetChar(Index: Integer; Character: Word);
```

Description
Modify one character in the paragraph. If the index is not valid this method will do nothing.

4.5.1.4 E) IWPParInterface.SaveToString

Applies to
IWPParInterface

Declaration
```
function SaveToString(const Format: WideString; OnlyChildren: WordBool):
WideString;
```

Description
Saves the contents of zthe paragraph to a string. Also see LoadFromString.

Category
Load and Save

4.5.1.4 F) IWPParInterface.ReplaceText

Applies to
IWPParInterface

Declaration
```
procedure ReplaceText(PosInPar: Integer; Len: Integer; const NewText:
WideString);
```

Description
Replaces a subtext with a different text.

4.5.1.4 G) IWPParInterface.LoadFromString

Applies to
IWPParInterface

Declaration
```
procedure LoadFromString(const Data: WideString; const Format: WideString; Mode:
Integer);
```

Description
Loads the contents of this paragraph from a string. This can be very useful in table cells.

Parameters	
Data	Text to be loaded
Format	Formatstring, i.e. "RTF", default is "AUTO"
Mode	Mode bits: 1: clear indents of loaded text 2: clear shading of loaded text 4: Load using current writing mode 8: Load text as children paragraphs (for table cells!)

Category
Load and Save

4.5.1.4 H) IWPParInterface.LoadFromFile

Applies to
IWPParInterface

Declaration
```
procedure LoadFromFile(const filename: WideString; const Format: WideString;
Mode: Integer);
```

Description
Loads the contents of this paragraph from a file. This can be very useful in table cells.

Parameters	
Filename	Name of the file to be loaded.
Format	Formatstring, i.e. "RTF", default is "AUTO"
Mode	Mode bits: 1: clear indents of loaded text 2: clear shading of loaded text 4: Load using current writing mode 8: Load text as children paragraphs (for table cells!)

Category
Load and Save

4.5.1.4 I) IWPParInterface.InsertNewObject

Applies to
IWPParInterface

Declaration
```
function InsertNewObject(Index: Integer; ObjType: Integer; HasClosing: WordBool;
ClosingOfPtr: Integer; CharAttrIndex: Integer; const Name: WideString; const
Command: WideString): Integer;
```

Description
This method creates a new object in the paragraph. The result is the ID of the object, it is != 0 if the insertion was successful. This method can also be used to create paired objects, such as hyperlinks, bookmarks or merge fields. You can use the result value in a subsequent call to this function as value for parameter ClosingOfPtr.

This procedure can be useful to create links and bookmarks in an event, such as OnEnumParOrStyle without having to change the cursor position. It can also be used to insert images.

Note: You can use GetCharObj to acquire an interface to manipulate the object.

Parameters	
Index	The position to insert the object. Use a large integer to append at the end of the paragraph.
ObjType	The object type.
HasClosing	If true a paired object will be created.
ClosingOfPtr	The ID of the start object. This object ID was returned by a previous call to InsertNewObject.
CharAttrIndex	The CharAttr index to be used for the new object placeholder.

Name	The name of the object.
Command	The command for this object, hyperlinks use this parameter as URL.

Category
Image Support

InsertNewObject Example

This event handler creates an image object in all paragraphs and loads an image file.

```
private void wpdllInt1_OnEnumParOrStyle(
 object Sender,
 bool IsControlPar,
 int StartPos,
 int Count,
 WPDynamic.IWPParInterface ParText,
 WPDynamic.IWPAttrInterface ParAttr,
 int EventParam,
 ref bool Abort)
{
// Make sure there is an image place holder at the start of the paragraph
WPDynamic.IWPTextObj obj = ParText.CharObj(0);
if ((obj==null)||  (obj.ObjType!=WPDynamic.TextObjTypes.wpobjImage))
{
 if(ParText.InsertNewObject(0,(int)WPDynamic.TextObjTypes.wpobjImage,false,
 0, 0, "", "")==0) obj=null;
 else obj = ParText.CharObj(0);
}
// Now update the image placeholder by loading a PNG file
if(obj!=null)
{
 obj.LoadFromFile("c:\\Test.PNG");
 obj.Width = obj.Contents_Width();
 obj.Height = obj.Contents_Height();
}
}
```

The test is updated with

```
wpdllInt1.Memo.EnumParagraphs(false,1);
wpdllInt1.Memo.ReformatAll(true,true);
```

Please note that you could clone the images if you have a global integer variable to hold the ID of the first image data object:

```
// Now update the image placeholder by loading a PNG file
if(obj!=null)
{
 if (objid==0)
 {
  obj.LoadFromFile("c:\\Test.PNG");
  objid = obj.GetContentsID();
 }
 else obj.SetContentsID(objid);
 obj.Width = obj.Contents_Width();
 obj.Height = obj.Contents_Height();
}
```

4.5.1.4 J) IWPParInterface.HasText

Applies to
IWPParInterface

Declaration
```
function HasText(const Text: WideString; CaseSensitive: WordBool): WordBool;
```

Description
This functions checks if a given text exists in this paragraph.

4.5.1.4 K) IWPParInterface.GetText

Applies to
IWPParInterface

Declaration
```
function GetText: WideString;
```

Description
This method retrieves the text of the paragraph as a string. Object placeholders will be suppressed.

Note, if you need to check if a paragraph is empty you can use `ParCommand(1,..)`.

4.5.1.4 L) IWPParInterface.GetSubText

Applies to
IWPParInterface

Declaration
```
function GetSubText(PosInPar: Integer; Len: Integer): WideString;
```

Description
You can use this method to extract a piece of the paragraph as a string.

4.5.1.4 M) IWPParInterface.GetChar

Applies to
IWPParInterface

Declaration
```
int GetChar(int Index);
```

Description
This function returns the character at a certain position in the current paragraph.

To read the character at the current position use
`Memo.CurrPar.GetChar(Memo.TextCursor.CPPosInPar);`

4.5.1.4 N) IWPParInterface.GetAllText

Applies to
IWPParInterface

Declaration
```
function GetAllText(AlsoNumbers: WordBool): WideString;
```

Description
This function returns the text of this paragraph and all the children. If the parameter AlsoNumbers has been set to true the numbering of numbered paragraphs will be added, too.

4.5.1.4 O) IWPParInterface.DeleteChar

Applies to
IWPParInterface

Use SetProp(1,1) to work with selected text.

Declaration
```
procedure DeleteChar(Index: Integer; Count: Integer);
```

Description
Delete N characters at a certain position.

4.5.1.4 P) IWPParInterface.CharObj

Applies to
IWPParInterface

Declaration
```
function CharObj(Index: Integer): IWPTextObj;
```

Description
This function provides a IWPTextObj interface to manipulate the object at a certain position in the paragraph. The result value is null if there is no object at that position.

4.5.1.5 Methods to manage the paragraph List

4.5.1.5 A) Low level move Methods

If you want to change the properties of the text using low level code you can use this methods to move to the next, previous, child or parent paragraph. In all cases the respective function returns false if the paragraph cannot be located.

The text is arranged in memory similar to objects in HTML code. Simple text does not use nesting, the paragraphs are siblings:
line 1
line 2
line 3

Tables use nesting. The table is the parent paragraph, it contains children which are rows. The rows have children which represents the cells. Cells may have children if they contain more than one paragraph. (The cell paragraph currently can also contain text. This helps to reduce the memory consumption of large tables)

```
<table>
  <tr>
   <td>Cell 1</td>
   <td>Cell 2</td>
   <td>Cell 3 - line one
      <p>Cell 3 line 2</p>
```

```
      <p>Cell 3 line 3</p>
    </td>
   </tr>
</table>
```

Note:
a) Please also see SetPtr which supports some constants to select the parent table or row.
b) Memo.CurrPar.Select..() **will** change the cursor position but Memo.ActiveText.CurrPar.Select...() will **not** change it!

Methods:

bool SelectNextPar(bool ChildrenToo);
Select the next sibling. If the parameter is true it will also move inside of the object tree.

bool SelectNextParGlobal(bool ChildrenToo);
Select the next sibling. If the parameter is true it will also move inside of the object tree. In contrast to SelectNextPar this method will move to the next RTFDataBlock if the current was the last object.

bool SelectPrevPar(bool ParentToo);
Select the previous sibling. If the parameter is true it will also move one level higher.

bool SelectChildPar()
Select the child paragraph - if there is one, otherwise the result is false.

bool SelectParentPar()
Select the parent paragraph - if there is one, otherwise the result is false.

bool SelectFirstSibling()
Select the first paragraph in the current level. This can be used to select the first row in a table or the first cell in a row.

bool SelectLastSibling()
Select the last paragraph in the current level. This can be used to select the last row in a table or the last cell in a row.

bool SelectSiblingNr(int n)
Select the n-th paragraph in the current level - if there is one, otherwise the result is false.

int GetSiblingNr()
Calculates the sibling number.

int GetSiblingCount()
Calculates the number of siblings in this level. This can be the number of rows or the number of cells.

also see:
IWPParInterface.SelectFirstPar
IWPParInterface.SelectLastPar
IWPParInterface.SelectFirstParGlobal

4.5.1.5 B) IWPParInterface.SwapWithPrevPar

Exchanges two paragraphs in the list of paragraphs.

4.5.1.5 C) IWPParInterface.SwapWithNextPar

Exchanges two paragraphs in the list of paragraphs.

4.5.1.5 D) IWPParInterface.SetPtr

Applies to
IWPParInterface

Declaration
```
bool SetPtr(int Paragraph);
```

Description
Makes the paragraph with the specified ID the 'current' paragraph. See method AppendChild for an example.

Instead of IDs the following constants can be used:
1: Select current cell. This is useful to select the cell a child paragraph is located within.
2: Select parent row.
3: Select parent table - for example to change the properties of a table directly.
4: Select parent cell, the cell which owns a nested table.
5: Select the Parent-Parent Table, the table which owns the parent Cell

If the selected object was not found, false is returned.

Please also see low "level move methods" which make it easy to move through a document without changing the cursor position!

Category
Lowlevel Paragraph IDs

4.5.1.5 E) IWPParInterface.SetParType

Applies to
IWPParInterface

Declaration
```
procedure SetParType(ParType: Integer);
```

Description
Modify the paragraph type. Currently You can use the values:
0 : Standard paragraph
6 : Table parent - holds rows
7 : Table row - holds cells
16: This is a special "sub-paragraph". Its text will be displayed as one line but the cursor can never move to the paragraph. Instead all new text paragraph are entered as children of this paragraph.

The .NET assembly defines the enum "ParagraphType".

Example: Create sub paragraphs

```
IWPMemo Memo = wpdllInt1.Memo;
IWPTextCursor Cursor = Memo.TextCursor;

Memo.Clear(true, true);

Cursor.InputParagraph(4, ""); // Append a paragraph
Memo.CurrPar.SetParType(16); // Sub Paragraph
```

```
            Memo.CurrPar.ParStrCommand(4, 0, "AAA"); // this is the name
            Memo.CurrPar.SetText("Headline 1",0); //CharAttr is ignored
            Memo.CurrPar.LoadFromString("{\\rtf1{Initial text for this \\b paragraph\
\b0 .}","",0);

            Cursor.InputParagraph(4, ""); // Append a paragraph
            Memo.CurrPar.SetParType(16); // Sub Paragraph
            Memo.CurrPar.ParStrCommand(4, 0, "BBB"); // this is the name
            Memo.CurrPar.SetText("Headline 2",0); //this is the displayed text
            Memo.CurrPar.LoadFromString("{\\rtf1{Initial text for this \\b paragraph\
\b0 .}","",0);

            Cursor.CPPosition = 0;
            Memo.ReformatAll();
```

You can use Memo.TextCommandStr(42, 4, "AAA") to move to a certain paragraph. Now you can use CurrAttr.LoadFromString or SaveToString.

Category
Table Support

4.5.1.5 F) IWPParInterface.SelectLastPar

This method jumps to the last paragraph in the layer.

4.5.1.5 G) IWPParInterface.SelectFirstParGlobal

Makes the IWPParagraphInterface work with the first paragraph in the document. This can be used to create loop all paragraphs using SelectNextGlobalPar. (see Low level move Methods)

4.5.1.5 H) IWPParInterface.SelectFirstPar

Makes the IWPParagraphInterface work with the first paragraph in this layer. This can be used to create loop all paragraphs using SelectNextPar. (see Low level move Methods)

4.5.1.5 I) IWPParInterface.GetPtrPrev

Applies to
IWPParInterface

Declaration
```
function GetPtrPrev: Integer;
```

Description
This is the ID of the previous sibling of the paragraph.

Category
Lowlevel Paragraph IDs

4.5.1.5 J) IWPParInterface.GetPtrParent

Applies to
IWPParInterface

Declaration
```
function GetPtrParent: Integer;
```

Description
This is the ID of the parent of the paragraph. You can use this value with SetPtr.

Category

Lowlevel Paragraph IDs

4.5.1.5 K) IWPParInterface.GetPtrNext

Applies to
IWPParInterface

Declaration
```
function GetPtrNext: Integer;
```

Description
This is the ID of the next sibling of the paragraph.

Category
Lowlevel Paragraph IDs

4.5.1.5 L) IWPParInterface.GetPtrChild

Applies to
IWPParInterface

Declaration
```
function GetPtrChild: Integer;
```

Description
This is the ID of the first child of this paragraph.

Category
Lowlevel Paragraph IDs

4.5.1.5 M) IWPParInterface.GetPtr

Applies to
IWPParInterface

Declaration
```
function GetPtr: Integer;
```

Description
This function returns the Integer ID of a paragraph. You can use SetPtr with a valid ID to make a different paragraph the "current".

Category
Lowlevel Paragraph IDs

4.5.1.5 N) IWPParInterface.GetParType

Applies to
IWPParInterface

Declaration
```
function GetParType: Integer;
```

Description
This function returns the paragraph type. Regular paragraphs use the type 0, tables 6 and table rows 7.

4.5.1.5 O) IWPParInterface.DuplicateEx

Applies to
IWPParInterface

Declaration
```
procedure DuplicateDuplicateEx(
   CopyText: Integer;
   CopyChildren: Integer;
   LinkBefore: Integer;
   Mode: Integer);
```

Description
Like "Duplicate" this method duplicates this paragraph and all its text and children. It has additional options.

CopyText - if <>0 all text will be copied, otherwise the text will be not copied.
CopyChildren - if <>0 the children (such as table rows or row cells) will be copied
LinkBefore - if <>0 the new paragraph will be inserted before the current

Mode: Only if bit 1 is set, the new paragraph will be made the current.

4.5.1.5 P) IWPParInterface.Duplicate

Applies to
IWPParInterface

Declaration
```
procedure Duplicate;
```

Description
This method duplicates this paragraph and all its text and children. It makes the new paragraph the current.

4.5.1.5 Q) IWPParInterface.DeleteParEnd

Applies to
IWPParInterface

Declaration
```
procedure DeleteParEnd;
```

Description
This method delets the end of the paragraph. This in fact combines the paragraph with the following paragraph. If the next paragraph cannot be appended (it is is a table) the function does nothing.

4.5.1.5 R) IWPParInterface.DeleteParagraph

Applies to
IWPParInterface

Declaration
```
procedure DeleteParagraph;
```

Description
This method deletes this paragraph and all children. It makes the next paragraph the current.

4.5.1.5 S) IWPParInterface.AppendNext

Applies to
IWPParInterface

Declaration
```
function AppendNext: Integer;
```

Description
This procedure appends a new paragraph after the current and makes it the current. If the current paragraph is a cell and you need to create a second paragraph inside the same cell please use AppendChild.

4.5.1.5 T) IWPParInterface.AppendChild

Applies to
IWPParInterface

Declaration
```
function AppendChild: Integer;
```

Description
This procedure creates a new child paragraph. If the current paragraph is a table paragraph the child will be a table row, if it is a table row, the child will be a cell. The result value is the ID of the new paragraph. It can be used in SetPtr to be manipulated by the IWPParInterface instance.

Example: Low level table creation:

```
IWPParInterface par= wpdllInt1.CurrPar;
int tbl=par.AppendNext();
par.SetParType((int)ParagraphType.Table);
for (int r= 0; r < 10; r++)
{
 par.SetPtr(par.AppendChild()); // row=current
 for (int c = 0; c < 5; c++)
 {
  if (c==0) par.SetPtr(par.AppendChild()); // First Cell
  else par.AppendNext(); // Other cells
  par.AppendText((r*10+c).ToString(),-1);
  par.Borders = 15;
 }
 par.SetPtr(tbl); // table=current
}
par.AppendNext();
par.AppendText("Text after Table",-1);
wpdllInt1.Reformat();
```

Category
Table Support

4.5.1.6 Other methods

4.5.1.6 A) Convert Utility function

The following methods can be used to create number values for the method ParASet or to convert the result of ParAGet.

a) Functions to work with font names

int ConvertFontnameToIndex(string fontface)

string ConvertIndexToFontname(int value)

used with this property ids:
WPAT_CharFont = 1
WPAT_NumberFONT = 40

b) Functions to work with color values

int ConvertColorToIndex(int RGB_Value)

string ConvertIndexToText(int value)

used with this property ids:

WPAT_CharColor = 8
WPAT_CharBGColor = 9
WPAT_UnderlineColor = 14
WPAT_NumberFONTCOLOR = 42
WPAT_BGColor = 50
WPAT_FGColor = 51
WPAT_BorderColor = 89
WPAT_BorderColorL = 72
WPAT_BorderColorT = 73
WPAT_BorderColorR = 74
WPAT_BorderColorB = 75

Reserved:
WPAT_BorderColorDiaTLBR = 76
WPAT_BorderColorDiaTRBL = 77
WPAT_BorderColorH = 80
WPAT_BorderColorV = 83
WPAT_BorderColorBar = 86

c) Functions to convert text values

int ConvertTextToValue(string SomeText)

int ConvertIndexToColor(int value)

used with this property ids:

WPAT_NumberTEXTB = 34
WPAT_NumberTEXTA = 35
WPAT_NumberTEXT = 36
WPAT_PAR_COMMAND = 151
WPAT_STYLE_NEXT_NAME = 181
WPAT_STYLE_BASE_NAME = 182

Tip: Some color index values are predefined - see list.

4.5.1.6 B) IWPParInterface.WidthTwips

This function returns the calculated width of this paragraphs in twips.

4.5.1.6 C) IWPParInterface.StartCPOffset

This function calculates the offset from the start of the text.

The result can be used in TextCursor.CPPosition.

To assign the offset to move the "CurrPar" interface use ParCommand with ComID=5.

4.5.1.6 D) IWPParInterface.SetProp

Applies to
IWPParInterface

Declaration

```
void SetProp(int ID, int Value);
```

Description

ID=1:
If You use Memo.CurrPar usually only the current paragraph, the paragraph the cursor is in, will be modified. **Certain methods can also modify all selected paragraphs**. To use this feature You first need to call SetProp with ID=1 and Parameter=1 to initialize the list of selected paragraphs. When done please don't forget to call SetProp with ID=1 and **Value=0** to clear the list. If not text was selected, at least the current paragraph will be added to the internal list.

We highlighted the methods which work on a paragraph list with this symbol:
Use SetProp(1,1) to work with selected text.

ID=2:
Add the current paragraph to the internal paragraph list (value is ignored). You need to call SetProp(1,0) to switch the "paragraph list" mode off.

4.5.1.6 E) IWPParInterface.GetProp

Applies to
IWPParInterface

Declaration
```
int GetProp(int ID);
```

Description
ID=1
Returns 1 if the interface is currently working on a paragraph list.

4.5.1.6 F) IWPParInterface.ParStrCommand

Applies to
IWPParInterface

Declaration
```
function ParStrCommand(ComID: Integer; param: Integer; const StrParam:
WideString): Integer;
```

Description
The following command IDs are currently used:

1: Convert a font name into a font index, use it with the code WPAT.CharFont and method ParASet. (Better use convert utility)

2: Convert a string into a number, use it with the code WPAT_NumberTEXTB and WPAT_NumberTEXTA. (Better use convert utility)

3: Convert a color string into a number, use it with the code WPAT_CharColor. (Better use convert utility)

4: Assigns the **name of this paragraph**. This is not the "cellname". It can be searched for with ID 10 and 11.

You can use Memo.TextCommandStr(42) to move the cursor to a paragraph with a certain name. (Use CurrPar.ParStrCommand with id 4 to set the name)

5: Assigns the "cell name" to the parent cell of this paragraph. Returns -1 if not in a table. (same as property CellName)

6: Assigns the "cell command" to the parent cell of this paragraph. Returns -1 if not in a table.

7: Assigns the name to the parent table of this paragraph. Returns -1 if not in a table. The table name can be used in MoveToTable.

8: Assigns a name to this paragraph.
Also see: IWPTextCursor.GetParName and SetParName.

9: A **paragraph style** can be set using property StyleName. For a more control apply the style using ParStrCommand(9, param, name) to set and apply a style. If bit 1 is set in "param", the character attributes, which are used by the selected style, are cleared, if bit 2 is set, the paragraph attributes are cleared. The style does not have to exist. The function returns its number if successful. (Use SetProp(1,1) to work with selected text.)
Also see: "Paragraph Styles".

10: **Locates** a paragraph with a given name (use ID 4 to set) and makes it the current. Returns a value>0 if the paragraph was found. If param <> 0, the search will be started with the first paragraph after the current.

11: Locates a paragraph with a given name - ignores cases.

12: Locates a paragraph with a given text and moves there.

Alternative to the commands 10 - 12: Memo.TextCommandStr(42).

Category
Attribute IDs

4.5.1.6 G) IWPParInterface.ParCommand

Applies to
IWPParInterface

Declaration
```
int ParCommand(int ComID, int param, int param2);
```

Description
Execute special commands for this paragraph, cell or table row.

ComID = 1: (check if empty)

Check wether the paragraph is empty. The result is 1 if it is empty, 0 if it is not, -1 the operation was not possible.
if param is 1 white space will be ignored.

param2 is a bitfield to ignore certain object types:
1: merge fields
2: hyperlinks
4: bookmarks
8: text protection codes (reserved)
16: span styles
32: custom codes (reserved)
64: text objects, such as a page number or a symbol object
128: references, used by the table of contents
256: reserved
512: reserved
1024: footnotes
2048: images
4096: horizontal line

ComID = 2: (height of paragraph)

This command reads the row height of the current table row in twips. It can also be used with cell paragraphs.

In case there is no table row, the result is -1.

param2 is used to enable the examination not only of the current row but of all rows in the current table:

0: return current row height
1: return the smallest row height in the table
2: return the tallest row height in the table
3: return the average row height in the table
else return row of all rows (sum)

Unless the complete table is modified the result value is the applied value in twips. In case the complete table is modified the result value is the count of modified rows.

ComID = 3: (height of paragraph)

This command applies the height value "param" to the current row.

param2 is used as a bitfield to control how the value is applied. If param2=0 the minimum height of the current row is changed.

1: Also set maximum height (WPAT_BoxMaxHeight)
2: Do not set minimum height (WPAT_BoxMinHeight)
4: Apply value to all rows in the current table
8: The value "param" is added to the current actual row height is used as input. This can be used to "lock in" the height.

ComID = 4: (paragraph state flags)

Work with the paragraph state flags:

param=0 clears the flag, param=1 sets the flag, param=-1 only reads the flag

param2 selects the flag:

0: paragraph is hidden (flag is also used by DeleteParWithCondition)
1: paragraph is hidden (alternative flag)
2: user flag 1
3: user flag 2
4: user flag 3
5: user flag 4
6: protects tab stops

You need to call ReformatAll to update the screen

ComID = 5: ("move" CurrPar interface)

locate the paragraph at Position "param". Returns -1 if a paragraph was not found (value too large), otherwise the position within the paragraph.

ComID = 6: (trim)

Trim function - removes whitespace from start and end of this paragraph / cell.

ComID = 7: (find attributes)

Search in current paragraph for certain character attributes.

Starts to search from position Param2 and returns -1 if nothing was found, or the position is stored in the high word and the length in the low word of the result text. (This restricts paragraphs not to exceed 64K characters)

Param selects the attribute in the high byte of the 32 bit parameter. (Actually You can use the first 16 WPAT codes here)

The low 3 bytes are the parameter.

The following values are useful in the highest byte:
1: The Font Nr
3: The font size * 100
7: The character style. The low byte selects the style.
8: The color nr.
Use the same values as in IWPAttrInterface.IncludeStyles and IWPAttrInterface.ExcludeStyles:

0 : Bold text. (C# wrapper defines enum element WPWRT.BOLD)
1 : Italic text. (C# wrapper defines enum element WPWRT.ITALIC)
2 : Underlined text. (C# wrapper defines enum element WPWRT.UNDERLINE)
3 : Strikeout text. (C# wrapper defines enum element WPWRT.STRIKEOUT)
4 : Text in super-script (C# wrapper defines enum element WPWRT.SUPERSCRIPT)
5 : Text in sub-script (C# wrapper defines enum element WPWRT.SUBSCRIPT)
6 : Hidden text, (C# wrapper: WPWRT.HIDDEN)
7 : Uppercase text. (C# wrapper: WPWRT.UPPERCASE)
8 : reserved.
9 : Lowercase text. (C# wrapper: WPWRT.LOWERCASE)
10 : Text which should be excluded from spellcheck (WPWRT.NOPROOF)
11 : Double strikeout (WPWRT.DBLSTRIKEOUT)
12 : reserved.
13 : protected text (WPWRT.PROTECTED)

The lower 3 bytes are the desired value. They have to match exactly, except for the character style comparison. In this case the given bits have to be set.

This method can also be used to search special flags. In this case "Param" may have this values
16+1: - locates hyphen marker for manual hyphenation
16+2: - locates text which was spell checked already
16+3: - locates text which was marked to misspelled by the spell-as-you-go feature
16+4: - the alternative misspell marker (green)
16+5: - reserved
16+6: - reserved
16+7: - highlighted text (on the fly highlights, not background color)
16+8: - marked for deletion

ComID = 8: (set attributes)

Assigns new attributes to part of the text. This is the counter part to the search command 7.
Param2 is the position and the length, high/low word.
Param selects the attribute.

1: The Font Nr
3: The font size * 100
7: Adds a character style like IWPAttrInterface.IncludeStyles
6: Removes a character styles like IWPAttrInterface.ExcludeStyles.

Example to ComID 7 and 8:
Make all bold text italic:

```
IWPMemo Memo = wpdllInt1.Memo;
IWPParInterface Par = Memo.CurrPar;
int a = 1;
int p = 0;
```

```
        while(a>0)
                {
                        a = Par.ParCommand(7, (7 << 24) + 0, p);
                        if (a>=0)
                        {
                                Par.ParCommand(8, (7 << 24) + 1, a); // Adds "Ital
                                Par.ParCommand(8, (6 << 24) + 0, a); // Removes "B
                                p = (a >> 16) + (a & 0xFFFF); // Calculate Next po
                        }
                }
        Memo.ReformatAll();
```

also see: EnumParagraphs

ComID = 8: (get pos on screen)

Param = position within paragraph (=posinpar)

If the result value = -1, the text at this position is not visible
Otherwise the highword = the x coordinate, the low word = y coordinate.

ComID = 9: (get X, y position of text in paragraph)

This Function can be used in the EnumParOrStyle event.

param = position in paragraph
param2 can have this values:
 0 : check if the character is visible. If not, the return 0, if yes return = 1
 1 : return x coordinate inside editor
 2 : return y coordinate inside editor
 3 : return x coordinate relative to screen
 4 : return y coordinate relative to screen

```
a = Par.ParCommand(7, (7 << 24) + 0, p);
if (a>=0)
{
   p = (a >> 16) + (a & 0xFFFF); // Calculate Next position
   if (Par.ParCommand(9, a >> 16, 0)==0)
   {
      Par.ParCommand(8, (8 << 24) + 1, a); // Selects red
   }
}
```

ComID = 10

Calculate the line number of the character at the position param. If param2<>0 it will favors the end of a line, otherwise the start.

ComID = 11

This Function returns the start offset of the line with number param (range 0..).

ComID = 12

This Function returns the length of the line with number param.

ComID = 13

This Function returns the value of various internal variables depending on param:

0 : pagenr - the logical page number the first character is printed on
1 : Number - the number which would be used when numbering the text
2 : NumGroup - the number level
3 : SectionID - the SectionID of the section this paragraph starts (IF it starts a section!)
4 : lastpagenr - the page number the last line of this paragraph was printed
5 : lasty - the last Y position
6 : LoadedCharAttr - the character attr index this paragraph defaults
7 : Level - the level of this paragraph (nesting!)
8 : SiblingCount - the count of siblings
9 : SiblingNr - the number within the siblings (i.e. column nr)
10 : IsFirstTextPar - 1 if this is the first paragraph, 0 if not
11 : par.IsLastTextPar - 1 if this is the last paragraph, 0 if not
12 : par.HasChildren - 1 if this paragraph has children, such as a table
13 : MinCHeight - height of smallest character
14 : IDPTR - a unique ID

15: CharCount but not counting objects

4.5.1.6 H) IWPParInterface.HeightTwips

This function returns the calculated height of this paragraphs in twips.

4.5.2 IWPDataBlock

Manage header and footer and text layers

Description
TextDynamic uses layers for header and footer texts. Layers are also used for text boxes and footnotes. This interface gives access to one layer element, called RTFDataBlock. This interface is used by Memo.ActiveText and also Memo.BlockAdd, Memo.BlockAppend and Memo.BlockFind.

An interface to this element can be useful to create header or footer in code. You can use the method Clear to erase the text. Use Delete to remove the layer completely.

You can append one paragraph using AppendParagraph. This can be useful to modify the text without having to move the cursor. Alternatively, to select the first paragraph with SelectFirstPar. In both cases the interfaces CurrPar and CurrParAttr can be used for low level text creation and formatting which does not require the change of the current cursor position. You can use the low level move methods to locate a different paragraph! (Make sure to call ReformatAll after the text has been updated.)

To move the cursor into the layer set property WorkOnText to true.

The property Kind is used to differentiate between header (DataBlockKind. wpIsHeader), footer (DataBlockKind.wpIsFooter), text boxes (DataBlockKind. wpIsOwnerSelected) and footnotes (DataBlockKind.wpIsFootnote). Also the body text uses the IWPDataBlock interface. It is the only layer which may span pages and its kind is DataBlockKind.wpIsBody.

So, if you need to check if the cursor is currently in the body text you can use a condition such as:

```
IWPDataBlock block = wpdllInt1.CurrMemo.ActiveText;
if (block!=null)&&(block.Kind == DataBlockKind.wpIsBody))
{
 // some code
}
```

This action starts the "manage header and footer" dialog:
wpaDiaManageHeaderFooter

Properties	Methods
ID	AppendParagraph
IsEmpty	Clear
Kind	Delete
Name	GetParPtrFirst
Range	GetParPtrLast
Readonly	SelectFirstPar
SectionID	
Text	
WorkOnText	
CurrPar	
CurrParAttr	

4.5.2.1 Properties

4.5.2.1 A) ID

Applies to
IWPDataBlock

Declaration
```
int ID;
```

Description
This is the ID of the data block. IDs are used to select certain data blocks in event OnGetSpecialText.

Do not mix up with SectionID.

4.5.2.1 B) IsEmpty

Applies to
IWPDataBlock

Declaration
```
bool IsEmpty;
```

Description

This property is true if the data block is completely empty.

4.5.2.1 C) Kind

Applies to
IWPDataBlock

Declaration
```
DataBlockKind Kind;
```

Description
The property Kind is used to differentiate between header (DataBlockKind.wpIsHeader), footer (DataBlockKind.wpIsFooter), text boxes (DataBlockKind.wpIsOwnerSelected) and footnotes (DataBlockKind.wpIsFootnote). Also the body text uses the IWPDataBlock interface. It is the only layer which may span pages and its kind is DataBlockKind.wpIsBody.

4.5.2.1 D) Name

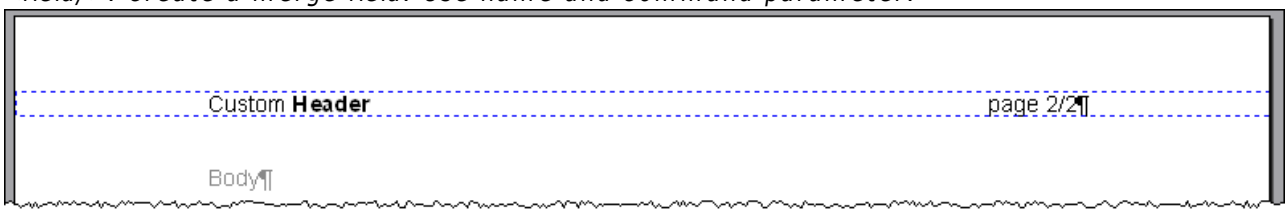
Applies to
IWPDataBlock

Declaration
```
string Name;
```

Description
The name can be used to locate a certain data block to be used as header or footer.

This code creates a new header text. The text which is assigned to "Text" may be RTF text or may contain HTML formatting tags and the following proprietary closed HTML tags: pagenr, pagecount, tab and field.

<pagenr/>: Print the current page number
<pagecount/>: Print the page count
<tab/>: Insert a tabchar and create a tabstop: left, right, center, decimal=twips-value
<field/>: create a merge field. Use name and command parameter.




```
// Create a custom header which will be printed only on page #2
IWPDataBlock block = wpdllInt1.Memo.BlockAdd(
     DataBlockKind.wpIsHeader,
     DataBlockRange.wpraNamed,
     "CUSTOM", 0);
block.Clear();
block.Text = "Custom <b>Header</b><tab right=9000/>page <pagenr>/<pagecount>";
```

This event handler selects the "custom" header for page number 2 only.
```
private void wpdllInt1_OnGetSpecialText(
  object Sender,
  int Editor,
  int PageNr,
  WPDynamic.DataBlockKind Kind,
```

```
  ref int SelectedID)
{
 if((Kind==DataBlockKind.wpIsHeader)&&(PageNr==2))
    {
  SelectedID=wpdllInt1.Memo.FindHeader(
     (int)DataBlockRange.wpraNamed,
     "CUSTOM" // name to find
     );
     }
}
```

4.5.2.1 E) Range

Applies to
IWPDataBlock

Declaration
```
DataBlockRange Range;
```

Description
The "range" is used by header or footer texts. Possible values are

```
wpraOnAllPages =0; // use on all pages
wpraOnOddPages =1; // use on odd pages (1,3,5,...)
wpraOnEvenPages=2; // use on even pages (2,4,6,...)
wpraOnFirstPage=3; // print on first page only
wpraOnLastPage=4; // print on last page only
wpraNotOnFirstAndLastPages=5; // Not on first or last pages
wpraNotOnLastPage=6; // on all but not on last page
wpraNamed=7; // never used. Use event OnGetSpecialText to select it
wpraIgnored=8; // Dont used
wpraNotOnFirstPage=9; // On all but not on first page
```

Notes:
Only the first 4 options are standard in RTF format.
Footnotes will use wpraOnAllPages and text boxes will use wpraNamed.

4.5.2.1 F) Readonly

Applies to
IWPDataBlock

Declaration
```
bool Readonly;
```

Description
If this property is true the layer cannot be edited.

4.5.2.1 G) SectionID

Applies to
IWPDataBlock

Declaration
```
int SectionID;
```

Description

This id, if <> 0, is used to select a header or footer for a certain section in the text.

Hint: Use IWPPageSize.GetProp(0) to retrieve the section ID for a certain section.

4.5.2.1 H) Text

Applies to
IWPDataBlock

Declaration
```
string Text;
```

Description
The property Text can be used to assign new text. Example see property name.

This property is write - only!

Category
Load and Save

4.5.2.1 I) WorkOnText

Applies to
IWPDataBlock

Declaration
```
bool WorkOnText;
```

Description
If this property is true the cursor (the insertion marker) is located within this layer. This property can be read and written.

VFP note: If you use WorkOnText to change the active text, please re-get Memo.TextCursor.

4.5.2.1 J) CurrPar

Applies to
IWPDataBlock

Declaration
```
IWPParInterface ParAttr
```

Description

This interface let you modify the current paragraph. The current paragraph is the paragraph which was appended last using the function AppendParagraph. The method SelectFirstPar will make the first paragraph the "current".

Note: Please don't forget to call ReleaseInt() with the returned interface at the end of your code.

4.5.2.1 K) CurrParAttr

Applies to
IWPDataBlock

Declaration
`IWPAttrInterface ParAttr`

Description

This interface let you modify the current paragraph. The current paragraph is the paragraph which was appended last using the function AppendParagraph. The method SelectFirstPar will make the first paragraph the "current".

Note: Please don't forget to call ReleaseInt() with the returned interface at the end of your code.

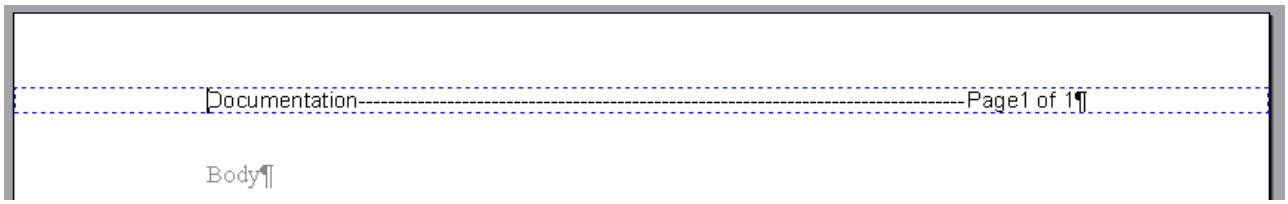
4.5.2.2 Methods

4.5.2.2 A) IWPDataBlock.AppendParagraph

Applies to
IWPDataBlock

Declaration
`IWPParInterface AppendParagraph();`

Description
This method appends a new paragraph to this text block and returns an IWPParInterface reference which can be used to manipulate this new paragraph. While it is easier to use the TextCursor methods to create a new header or footer, this method can be ideal to on the fly modify the text in header or footer since the cursor position does not have to be changed.



This header was created using the following C# code.

```
IWPAttrInterface currattr = wpdllInt1.CurrAttr;
// Create a custom header which will be printed only on page #2
IWPDataBlock block = wpdllInt1.Memo.BlockAdd(
     DataBlockKind.wpIsHeader,
     DataBlockRange.wpraOnFirstPage, "", 0);
block.Clear();
IWPParInterface par= block.AppendParagraph();
currattr.Clear();
currattr.SetFontface("Times New Roman");
currattr.SetFontSize(12);
par.AppendText("Documentation\tPage",-1); // use default attribute
par.InsertNewObject(1000,(int)TextObjTypes.wpobjTextObject,false,0,-1, "PAGE", '
par.AppendText(" of ",-1);
par.InsertNewObject(1000,(int)TextObjTypes.wpobjTextObject,false,0,-1, "NUMPAGES
// now also set tabstops
par.TabAdd(9000, 1, 3, 0);
```

If you need to create a second line you can use the API AppendNext:

```
par.AppendNext();
par.AppendText("Line 2",-1);
```

It is also possible to create a new table. (The IWPParInterface is a working interface, it always works on one paragraph. This paragraph can also be selected using the method SetPtr. AppendNext will automatically select the new paragraph.)

```
int tbl = par.AppendNext();
par.SetParType((int)ParagraphType.Table);
// We create a row object
int row = par.AppendChild();
// And now create 3 cells
par.SetPtr(row);
par.SetPtr(par.AppendChild());
par.AppendText("Cell A1",-1);
par.SetPtr(row);
par.SetPtr(par.AppendChild());
par.AppendText("Cell A2",-1);
par.SetPtr(row);
par.SetPtr(par.AppendChild());
par.AppendText("Cell A3",-1);
// Tip: to create a second row use
// par.SetPtr(tbl); and then create another row and cells
```

Please note that it is not possible to change the text in custom paint events.

Category
Table Support

4.5.2.2 B) IWPDataBlock.Clear

Applies to
IWPDataBlock

Declaration
```
void Clear();
```

Description
This procedure clears the text in this block.

4.5.2.2 C) IWPDataBlock.Delete

Applies to
IWPDataBlock

Declaration
```
void Delete();
```

Description
To delete the entire block use the procedure Delete.

4.5.2.2 D) IWPDataBlock.GetParPtrFirst

Applies to
IWPDataBlock

Declaration
```
int GetParPtrFirst()
```

Description
This is the ID of the first paragraph in this text block.

4.5.2.2 E) IWPDataBlock.GetParPtrLast

Applies to
IWPDataBlock

Declaration
```
int GetParPtrFirst()
```

Description
This is the ID of the last paragraph in this text block.

4.5.2.2 F) IWPDataBlock.SelectFirstPar

Applies to
IWPDataBlock

Declaration
```
bool SelectFirstPar()
```

Description
This method selects the the first paragraph in this text block to be the "current". You can then use CurrPar and CurrParAttr to manipulate this paragraph. To move to a different paragraph use, for example, CurrPar.SelectNextPar().

If the text block is empty the function will return false. You can use AppendParagraph to create a new paragraph.

4.5.3 IWPTextObj

Change field, object and image properties

Description
The interface IWPTextObject allows it to load the text of mail merge fields, to position image objects, to select text and move the cursor.

It is a low level interface which is usually not required to create text.

But to manipulate the object which is currently selected CurrSelObj or to check objects inside an event (OnEnumTxtObj) it can be very useful.

The property EmbeddedText allows it to read and modify the text within paired objects, such as hyperlinks, bookmarks and mail merge fields.

4.5.3.1 Properties

4.5.3.1 A) Command

Applies to
IWPTextObj

Declaration
```
string Command;
```

Description
This is the command property of this object. It has different meanings, hyperlinks store the URL, merge fields an optional formula.

4.5.3.1 B) Contents_Filename

Applies to
IWPTextObj

Declaration
```
string Contents_Filename;
```

Description
This is the file name of the image object linked to this object.

4.5.3.1 C) EmbeddedText

Applies to
IWPTextObj

Declaration
```
string EmbeddedText;
```

Description
This is the text which is wrapped by this object and its sibling end object. This only works with paired objects, such as hyperlinks, merge fields and bookmarks.

Note: You may read and modify this property!

4.5.3.1 D) Frame

Applies to
IWPTextObj

Declaration
```
int Frame;
```

Description
If the object is an image you can specify a frame:
0 : no frame
1 : single frame
2 : think frame

4.5.3.1 E) Height

Applies to
IWPTextObj

Declaration
```
int Height;
```

Description
This is the height of the object in twips (=1/1440 inch). It is only used by image and TextBox

objects.

4.5.3.1 F) IntParam

Applies to
IWPTextObj

Declaration
```
int IntParam;
```

Description
Integer property can be used to force a certain display width of the object in twips. (=1/1440 inch)

4.5.3.1 G) Mode

Applies to
IWPTextObj

Declaration
```
int Mode;
```

Description
This are the Mode bits for this object. The Value 2 marks edit fields to be editable. The other bits are only used by movable image objects:
4 : Lock position
8 : Allows sizing only "aspect ratio"
16: Disable size changes
32: Position at right side of page
64: Center object horizontally
128: Position in Margin.

4.5.3.1 H) Name

Applies to
IWPTextObj

Declaration
```
string Name;
```

Description
This is the name of the objects, for example the fieldname of a mailmerge field.

4.5.3.1 I) ObjType

Applies to
IWPTextObj

Declaration
```
TextObjTypes ObjType;
```

Description

The type of the object. The property is readonly, You can use _SetObjType to change the type.

0=text object,
1=merge field,
2=hyperlink,
3=bookmark,
7=text object,
8=page reference,
11=footnote,
12=image or text box and
13 for horizontal lines.

Please note: "Bookmarks" are NOT the kind of bookmarks used by PDF. This are invisible marks which can be used to locate certain text in a document. They are implemented as object pairs (like hyperlinks and merge fields) and so can contain text. To mark text to be exported as a PDF bookmark use the WPAT code ParIsOutline.

4.5.3.1 J) Params

Applies to
IWPTextObj

Declaration
```
string Params;
```

Description
This is the default text displayed by a textobject field.

4.5.3.1 K) PositionMode

Applies to
IWPTextObj

Declaration
```
int PositionMode;
```

Description
Image objects can be positioned relatively to a character (0), to the paragraph (1) and to the current page (2).

With the modes 1 and 2 the properties RelX and RelY are used to move the image using twips offset values.

4.5.3.1 L) RelX

Applies to
IWPTextObj

Declaration
```
int RelX;
```

Description
Used to position a movable image (PositionMode =1 or 2). It is the horizontal offset in twips.

4.5.3.1 M) RelY

Applies to

IWPTextObj

Declaration
```
int RelY;
```

Description
Used to position a movable image (PositionMode =1 or 2). It is the vertical offset in twips.

4.5.3.1 N) StyleName

Applies to
IWPTextObj

Declaration
```
string StyleName;
```

Description
Some paired objects (SPAN and Hyperlink) can use an attached patragraph style to format the embedded text. This property contains the name of this style.

Category
Paragraphstyle Support

4.5.3.1 O) Width

Applies to
IWPTextObj

Declaration
```
int Width;
```

Description
This is the width of the object in twips (1/1440 inch). It is only used by images.

4.5.3.1 P) wpcss

Applies to
IWPTextObj

Declaration
```
string wpcss;
```

Description
Read and write the properties as WPCSS string.

4.5.3.1 Q) Wrap

Applies to
IWPTextObj

Declaration
```
int Wrap;
```

Description
Used to control the text wrap around a movable image (PositionMode =1 or 2).
0 : automatic, wrap on the wider side

1 : wrap on left side
2 : wrap on right side
3 : do not wrap at all - object is painted over text.
4 : wrap on both sides

4.5.3.2 Methods

4.5.3.2 A) IWPTextObj._SetObjType

Applies to
IWPTextObj

Declaration
```
procedure _SetObjType(Value: Integer);
```

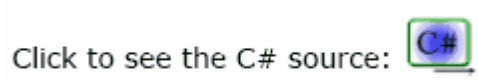
Description
This method sets the type of the object. Use with care.

The following types are possible:
0=text object, 1=merge field, 2=hyperlink, 3=bookmark, 7=text object, 8=page reference, 11=footnote, 12=image or text box and 13 for horizontal lines.

Tip:
This method can be used to convert an embedded image into a special image type which can also store attached data when exported to PDF.
If you insert a new image better use TextCursor.InputEmbeddedData.
Use _SetObjType(**100**) to make this conversion. This will only work if the ObjType is currently 12. After this change you can use LoadFromFile to load the contents of this object. When the PDF export is performed the image will be clickable. Acrobat Reader can extract the embedded data.

Example (C#):

```
td.Memo.TextCursor.InputString("Click to see the source: ",0);
tdf.Memo.TextCursor.InputImage("c:\\path\\cs_logo.png",0);
td.Memo.CurrObj._SetObjType(100);
td.Memo.CurrObj.ScaleSize(567,0,0); // Scale to width = 1 cm
td.Memo.CurrObj.LoadFromFile("c:\\path\\WebForm1.aspx.cs");
```



4.5.3.2 B) IWPTextObj.Clear

Applies to
IWPTextObj

Declaration
```
procedure Clear;
```

Description
This method clears the text of mergefields, hyperlinks and bookmarks. It also clears the contents of footnotes and textboxes (premium edition).

4.5.3.2 C) IWPTextObj.Contents_Edit

Applies to
IWPTextObj

Declaration
```
function Contents_Edit: WordBool;
```

Description
This method selects the edit mode for footnotes and text boxes.

4.5.3.2 D) IWPTextObj.Contents_Height

Applies to
IWPTextObj

Declaration
```
function Contents_Height: Integer;
```

Description
This is the physical height of the loaded image.

4.5.3.2 E) IWPTextObj.Contents_LoadFromFile

Applies to
IWPTextObj

Declaration
bool Contents_LoadFromFile(string filename);

This method can be only used with image objects which have been already created. It will update the image from the provided image file.

4.5.3.2 F) IWPTextObj.Contents_SaveToFile

Applies to
IWPTextObj

Declaration
```
function Contents_SaveToFile(const filename: WideString): WideString;
```

Description
This method can be used to save an image to file. It returns the name of the file which was actually written.

4.5.3.2 G) IWPTextObj.Contents_Width

Applies to
IWPTextObj

Declaration
```
function Contents_Width: Integer;
```

Description

This is the physical width of the loaded image.

4.5.3.2 H) IWPTextObj.DeleteObj

Applies to
IWPTextObj

Declaration
```
procedure DeleteObj;
```

4.5.3.2 I) IWPTextObj.GetContentsID

Applies to
IWPTextObj

Declaration
```
function GetContentsID: Integer;
```

Description
Retrieves the ID of the image object which is attached to the object placeholder. If no image is attached the ID will be 0.
You can use SetContentsID to assign the object also to another placeholder which clones the image.

4.5.3.2 J) IWPTextObj.GetEmbText

Applies to
IWPTextObj

Declaration
```
procedure GetEmbText(const Format: WideString; var Data: WideString);
```

Description
This property can be used to retrieve the contents as formatted text. It is usually used with merge fields.

4.5.3.2 K) IWPTextObj.GetFieldProp

Applies to
IWPTextObj

Declaration
```
function GetFieldProp(ID: Integer; var Value: WideString): WordBool;
```

4.5.3.2 L) IWPTextObj.GetParentParPos

Applies to
IWPTextObj

Declaration
```
function GetParentParPos: Integer;
```

Description

This is the position of the object inside the parent paragraph.

4.5.3.2 M) IWPTextObj.GetParentParPtr

Applies to
IWPTextObj

Declaration
```
function GetParentParPtr: Integer;
```

Description
This function retrieves a low level reference to the paragraph this object is located within.

4.5.3.2 N) IWPTextObj.GetProp

Applies to
IWPTextObj

Declaration
```
function GetProp(ID: Integer): WideString;
```

Description
This method is used to set the properties set by SetProp.

4.5.3.2 O) IWPTextObj.GetPtr

Applies to
IWPTextObj

Declaration
```
function GetPtr: Integer;
```

Description
This method retrieves a low level pointer to this object. This pointer can be carefully(!) used with other methods which work with "Ptr" values, such as MakeEndTag.

4.5.3.2 P) IWPTextObj.LoadFromFile

Applies to
IWPTextObj

Declaration
```
function LoadFromFile(const filename: WideString): WordBool;
```

Description
This method loads the contents of this object from a file.

Parameters	
Filename	The path to an image file (JPEG, PNG, BMP, EMF).

Note: When working with a placeholder object which embeds data into a PDF file (See InputEmbeddedData) this method updates the embedded data, not the image! In this case the file may be in RTF, C# and various other formats!

Category
Load and Save
Image Support

4.5.3.2 Q) IWPTextObj.LoadFromStream

Applies to
IWPTextObj

Declaration
```
function LoadFromStream(const FileExt: WideString; const Stream: IUnknown): WordBool;
```

Description
This method loads the contents of this object from a IStream or IWPStream. This is method is used to load image data into image objects. **.NET: The stream converter Stream2WPStream must be re-created for each load and save operation!**

Example:

```
System.IO.Stream str = new System.IO.MemoryStream();
// ... load data into "str" ... and then load it into the object
wpdllint1.Memo.CurrObj.LoadFromStream("PNG", new WPDynamic.Stream2WPStream(str));
```

Parameters	
FileExt	This is the extension string to describe the data format, for example JPG, EMF or BMP. When you use this method to load data into an PDF attachment container (_SetObjType) use a specify a filename.
Stream	This is a reference to an IStream or IWPStream interface.

Note: When working with a placeholder object which embeds data into a PDF file (See InputEmbeddedData) this method updates the embedded data, not the image! In this case the file may be in RTF, C# and various other formats!

Category
Load and Save

4.5.3.2 R) IWPTextObj.MakeEndTag

Applies to
IWPTextObj

Declaration
```
function MakeEndTag(PtrOfStartTag: Integer): WordBool;
```

Description
This method can be used to create object pairs, such a merge fields and hyperlinks. Usually you will not have to use this method, rather use high level functions such as InputHyperlink.
To create an object pair call Obj.MakeEndTag(PtrOfStartTag). PtrOfStartTag must have been retrieved using a call to Obj.GetPtr with a different IWPTextObj interface.

4.5.3.2 S) IWPTextObj.MoveCursor

Applies to
IWPTextObj

Declaration
```
procedure MoveCursor(Mode: Integer);
```

Description
This method moves the cursor near the

Parameters	
Mode	0 : Move before object 1 : Move after object 2 : Move before start tag - if paired object such as a field 3 : Move after start tag 4 : Move before end tag 5 : Move after end tag

4.5.3.2 T) IWPTextObj.MoveToPage

Applies to
IWPTextObj

Declaration
```
function MoveToPage(PageNr: Integer; X: Integer; Y: Integer): WordBool;
```

Description
not used yet.

4.5.3.2 U) IWPTextObj.ObjCommand

Applies to
IWPTextObj

Declaration
bool ObjCommand(int ID, int param,string StrParam);

Currently this commands are defined:

0 : Move cursor to this object
1 : Move cursor to the start tag
2 : Move Cursor to end tag

The Result is TRUE if the operation was successful.

4.5.3.2 V) IWPTextObj.ScaleSize

Applies to
IWPTextObj

Declaration
```
void ScaleSize(int BestWidth, int BestHeight, int Percent);
```

Description
This method can be used scale an image object.

Parameters	
BestWidth	The required width in twips - or 0
BestHeight	The required height in twips - or 0. BestWidth and BestHeight can also be both specified and will then define the respective maximum for width and height.
Percent	The percentage value to enlarge or shrink the image.

4.5.3.2 W) IWPTextObj.Select

Applies to
IWPTextObj

Declaration
```
void Select(int Mode);
```

Description
This methods selects the object.

Parameters	
Select	0 : only the object will be selected. 1 : select the embedded text (if paired object) 2 : also select the object start and end markers.

4.5.3.2 X) IWPTextObj.SetContentsID

Used for image cloning

Applies to
IWPTextObj

Declaration
```
function SetContentsID(ID: Integer): WordBool;
```

Description
GetContentsID retrieves the ID of the image object which is attached to the object placeholder. If no image is attached the ID will be 0.
You can use SetContentsID(id) to assign the object also to another placeholder which clones the image.

Parameters	
ID	May be 0 or a valid image data ID

Returns
True if assignment was ok, false if ID is not valid.

Category
Image Support

4.5.3.2 Y) IWPTextObj.SetEmbText

Applies to
IWPTextObj

Declaration
```
procedure SetEmbText(const Data: WideString; const Format: WideString);
```

Description
This method replaces the embedded text with formatted text. Unlike property EmbeddedText also HTML and RTF code can be used!

4.5.3.2 Z) IWPTextObj.SetFieldProp

Applies to
IWPTextObj

Declaration
```
procedure SetFieldProp(Id: Integer; const Text: WideString);
```

4.5.3.2 AA) IWPTextObj.SetProp

Applies to
IWPTextObj

Declaration
```
procedure SetProp(ID: Integer; const Value: WideString);
```

Description

With **ID<=0 custom string properties** can be set in this object.

-ID is used as ID of the specific custom string. Please use adjacent numbers.

Alternatively these flags can be set:
id = 1 : Set the mode [wpobjDrawAsRect]
2 : Set the mode [wpobjDrawAsText]
3 : Set the mode [wpobjWithinEditable]
4 : Set the mode [wpobjWithinProtected]
5 : Set the mode [wpobjPositionAtRight]
6 : Set the mode [wpobjPositionAtCenter]
7 : Set the mode [wpobjPositionInMargin]
8 : Set the mode [wpobjLockedPos]
9 : Set the mode [wpobjDisableAutoSize]
10 : Set the mode [wpobjSizingDisabled]
11 : Set the mode [wpobjSizingAspectRatio]
12 : Set the mode [wpobjObjectUnderText]
13 : Set the mode [wpobjReadSourceFromEmbeddedText]
14 : Set the mode [wpobjUseForMailmerge]
To set the flag use the value "1", to clear it use the value "0".

32: Set the value of the internal property "CParam"

33: Set the value of the internal property "IParam" and expect an integer value

34: Set the value of the internal property "IParam" and expect a color value. (You can set the color of a horizontal line)

4.5.3.2 AB) IWPTextObj.SetPtr

Applies to
IWPTextObj

Declaration
```
function SetPtr(ObjectPtr: Integer): WordBool;
```

Description
This methods attaches this interface to a different object instance. Use with care.

4.5.3.2 AC) IWPTextObj.ShowHint

Displays a hint window near this object.

Applies to
IWPTextObj

Declaration
```
procedure ShowHint(Mode: Integer; const Text: WideString);
```

Description
This method can be used to display hint over (mode=1) or under (Mode=0) the object. If mode =2 the hint window will be aligned to the base line of the paragraph. The hint will be visible for 2 seconds.

4.5.4 IWPNumberStyle

Interface to modify a numbering style.

Description
This interface is provided by GetNumberStyle. It can be used to create and modify numbering styles.
GetNumberStyle can also be used to add new number styles. To do so pass -1 als first parameter ID.

The following C# code will initialize legal outline numbering and also create some example text.

```
IWPMemo memo = wpdllInt1.CurrMemo;
IWPParInterface par = memo.CurrPar;
IWPTextCursor cursor = memo.TextCursor;
IWPNumberStyle style;
// Create legal numbering outline
for (int i = 1; i < 10; i++)
{
 style = memo.GetNumberStyle(-1,0,i);
 if(style!=null)
 {
  style.Mode = 3; // arabic
  style.TextA= "."; // after text = "."
  style.TextB= "";  // before text = ""
  style.Indent = 720; // 1/2 inch indent
  par.NumberLevel = 1;
  cursor.InputParagraph(0,"");
  par.SetText("Level B",0);
  par.NumberLevel = 2;
  cursor.InputParagraph(0,"");
  par.SetText("Level C",0);
  par.NumberLevel = 3;
  cursor.InputParagraph(0,"");
  par.SetText("continued C",0);
  par.NumberLevel = 3;
  cursor.InputParagraph(0,"");
  par.SetText("not numbered",0);
  par.NumberLevel = 0;
  cursor.InputParagraph
```

Result:

1. Level A
1.1. Level B
1.1.1. Level C
1.1.2. continued C
not numbered
1.2. Level B
2. Level A

```
  style.LegalNumbering = true;
//1.1.1 mode
 }
}
// Create numbered text
cursor.InputParagraph(0,"");
par.SetText("Level A",0);
// Optional
// par.IndentLeft = 720;
// par.IndentFirst= -720;
(0,"");
par.SetText("Level B"
,0);
par.NumberLevel = 2;
cursor.InputParagraph
(0,"");
par.SetText("not
numbered",0);
par.NumberLevel = 0;
par.SetText("Level A"
,0);
par.NumberLevel = 1;
// Reformat an paint
memo.ReformatAll(
false,true);
```

4.5.4.1 Properties

4.5.4.1 A) Color

Applies to
IWPNumberStyle

Declaration
`int Color;`

Description
The color for the number text (or bullets) as RGB value.

4.5.4.1 B) Font

Applies to
IWPNumberStyle

Declaration
`string Font;`

Description
The font for numbering or bullets. If empty use the paragraph phont.

4.5.4.1 C) Group

Applies to
IWPNumberStyle

Declaration
`int Group;`

Description
The outline group, 0 or 1.

If the value is 0 this number style is not part of a group, if it is 1 this is an outline style. Up to 9 levels can be defined in an outline style group.

4.5.4.1 D) ID

Applies to
IWPNumberStyle

Declaration
```
int ID;
```

Description
The ID used for the paragraph property WPAT_NumberStyle.

4.5.4.1 E) Indent

Applies to
IWPNumberStyle

Declaration
```
int Indent;
```

Description
The width reserved for the number or the bullet.

4.5.4.1 F) LegalNumbering

Applies to
IWPNumberStyle

Declaration
```
bool LegalNumbering;
```

Description
If true use legal numbering: 1.1.1

4.5.4.1 G) Level

Applies to
IWPNumberStyle

Declaration
```
int Level;
```

Description
0 or the level in an outline group between 1 and 9.

4.5.4.1 H) Mode

Applies to
IWPNumberStyle

Declaration
```
int Mode;
```

Description
The numbering mode:

0 : no numbering
1 : bullets
2 : circles
3 : arabic numbering 1,2,3
4 : captital roman
5 : roman
6 : capital latin
7 : latin

4.5.4.1 I) Size

Applies to
IWPNumberStyle

Declaration
```
float Size;
```

Description
The font size for the numbering, 0 to use paragraph attribute.

4.5.4.1 J) TextA

Applies to
IWPNumberStyle

Declaration
```
string TextA;
```

Description
The text **after** the number.

Hint: To use a unicode value, for example a dash, You can assign "\u2014"

4.5.4.1 K) TextB

Applies to
IWPNumberStyle

Declaration
```
string TextB;
```

Description
The text **before** the number. You can use it to define the bullet symbol. For bullets also change Font.

Hint: To use a unicode value, for example a dash, You can assign "\u2014"

4.5.4.1 L) WPCSS

Applies to
IWPNumberStyle

Declaration
```
string WPCSS;
```

Description
Get and set the properies as WPCSS string.

4.5.4.2 Methods

4.5.4.2 A) IWPNumberStyle.ADel

Applies to
IWPNumberStyle

Declaration
```
procedure ADel(WPAT_Code: Integer);
```

Description
Low level method to delete a property referenced by a WPAT id.

4.5.4.2 B) IWPNumberStyle.AGet

Applies to
IWPNumberStyle

Declaration
```
function AGet(WPAT_Code: Integer; var Value: Integer): WordBool;
```

Description
Low level method to read a property referenced by a WPAT id.

4.5.4.2 C) IWPNumberStyle.ASet

Applies to
IWPNumberStyle

Declaration
```
procedure ASet(WPAT_Code: Integer; Value: Integer);
```

Description
Low level method to modify a property referenced by a WPAT id.

4.5.4.2 D) IWPNumberStyle.Command

Applies to
IWPNumberStyle

Declaration
```
function Command(ID: Integer; Param: Integer): Integer;
```

Description
ID=1: Change locked state of style.
bit 1: Clear does not delete this style.
bit 2: Load will not overwrite this style.
bit 3: Always saved in WPT format, even if not used in document.

4.5.4.2 E) IWPNumberStyle.DeleteStyle

Applies to
IWPNumberStyle

Declaration

```
procedure DeleteStyle;
```

Description
Delete this style. The interface must not be used afterwards.
We recommend to use GetNumberStyle(ID,-1000,level) to delete styles.

4.5.4.2 F) IWPNumberStyle.SelectStyle

Applies to
IWPNumberStyle

Declaration
```
procedure SelectStyle;
```

Description
Make the style the current style referenced by CurrStyle.

4.6 Document Properties (page size, labels)

4.6.1 IWPPageSize

Interface to update Pagesize

Description
This interface is used to change the width and height of the page. All integer properties use twips. One twip is 1/1440 of an inch. If you work with centimeters, you can use this formula to convert centimeter into twips:
1 twip = 1 cm * 2.54 / 1440.

While mailing label print and preview is activated (see Memo.LabelDef) the page size defined using PageSize or in the OnMeasurePage is overridden.

The editor can be switched into the WordWrap mode - then the text is formatted to exactly fit into the window horizontally.

To make the current page size the default for new documents call "MakeDefault" or the method Memo.TextCommandStr(19,0,"");

```
wpdllInt1.Memo.TextCommandStr(19, 0, "");
```

Tip: To create a new section in the text you can use IWPTextCursor.InputSection. To select the current section use InputSection(-1).

```
IWPPageSize sect;
sect = wpdllInt1.TextCursor.InputSection(1);
sect.Landscape = true;
wpdllInt1.ReleaseInt(sect);
```

Using GetProp(0) the section ID can be retrieved. This ID can be used in Memo.BlockAdd as last parameter to create a header/footer for a certain section.

Note: The select flag (use SetProp to set it and GetProp to read it) will be

automatically set when a property is changed.

Properties	Methods
BottomMargin	GetProp
Landscape	SetPageWH
LeftMargin	SetProp
MarginFooter	
MarginHeader	
MarginMirror	
PageHeight	
PageWidth	
RightMargin	
TopMargin	

PAR-MAXLENGTH-Tip: You can use the method Memo.TextCommand(4,N,0) to specify that a maximum of N characters should be printed in one line.

4.6.1.1 Properties

4.6.1.1 A) BottomMargin

Applies to
IWPPageSize

Declaration
```
int BottomMargin;
```

Description
The margin area at the bottom of the page.

4.6.1.1 B) Landscape

Applies to
IWPPageSize

Declaration
```
bool Landscape;
```

Description
The orientaion of the page.

4.6.1.1 C) LeftMargin

Applies to
IWPPageSize

Declaration
```
int LeftMargin;
```

Description

The margin area at the left side of the page.

4.6.1.1 D) MarginFooter

Applies to
IWPPageSize

Declaration
```
int MarginFooter;
```

Description
This is the margin between the border of the page and the footer text.

4.6.1.1 E) MarginHeader

Applies to
IWPPageSize

Declaration
```
int MarginHeader;
```

Description
This is the margin between the border of the page and the header text.

4.6.1.1 F) MarginMirror

Applies to
IWPPageSize

Declaration
```
bool MarginMirror;
```

Description
If this property is true the left and right margin will be swapped on every otrher page.

4.6.1.1 G) PageHeight

Applies to
IWPPageSize

Declaration
```
int PageHeight;
```

Description
The page height measured in twips.

4.6.1.1 H) PageWidth

Applies to
IWPPageSize

Declaration
```
int PageWidth;
```

Description
The page width measured in twips.

4.6.1.1 I) RightMargin

Applies to
IWPPageSize

Declaration
```
int RightMargin;
```

Description
The margin area at the right side of the page.

4.6.1.1 J) TopMargin

Applies to
IWPPageSize

Declaration
```
int TopMargin;
```

Description
The margin area at the top of the page.

4.6.1.2 Methods

4.6.1.2 A) IWPPageSize.GetProp

Applies to
IWPPageSize

Declaration
```
int GetProp(int ID);
```

Description

Use **ID=0** to retrieve the "Section ID" of this section.

This section id can be used in Memo.BlockAdd as last parameter to create a header/footer for a certain section. Also see SectionID.

Use **ID=1** to retrieve a bit field which tells the editor which properties are controlled (selected) by this section object.

1 : PageSize,
2 : Margins
4 : Tab Default
8: Page Mirror
16: Page Numbering Mode (reserved)
32: Select Header and footer
64: Reset numbers of Outline
128: Reset Page number

Use **ID=2** to retrieve the page numbering mode for this section

1=Default, 2=Arabic, 3=Roman lowercase, 4=Roman uppercase

4.6.1.2 B) IWPPageSize.SetProp

Applies to
IWPPageSize

Declaration
```
procedure SetProp(Id: Integer; Value: Integer);
```

Description
Use **ID=1** to set a bitfield which tells the editor which properties are controlled (selected) by this section object.

1 : PageSize (Landscape, PageWidth and PageHeight),
2 : Margins
4 : Tab Default
8: Page Mirror
16: Page Numbering Mode (reserved)
32: Select Header and footer
64: Reset numbers of Outline
128: Reset Page number

Note: The bits 1-4 are automatically set when the respective property is changed!

Use **ID=2** to set the page numbering mode for this section

1=Default, 2=Arabic, 3=Roman lowercase, 4=Roman uppercase

4.6.1.2 C) IWPPageSize.SetPageWH

Applies to
IWPPageSize

Declaration
```
function SetPageWH(WidthTW: Integer; HeightTW: Integer; MargL: Integer; MargR:
Integer; MargT: Integer; MargB: Integer): Boolean;
```

Description
This method can be used to set width, height and margins in one line of code. For properties which should not be updated pass -1.

4.6.1.2 D) IWPPageSize.MakeDefault

Call this method to make this page definition the default. It will be applied, when the editor is cleared.

4.6.1.2 E) IWPPageSize.ReadDefault

Call this method to initialize the page definition with the default values.

4.6.2 IWPPageSizeList

You can use this property to install different page sizes for each page.

For Example If you need the first page to be extra lager. In this case just create 1 entry for the first page and another for all subsequent pages.

You can use the method Clear to remove all page size elements.

Use Add(int Width, int Height) or AddEx(int Width, int Height, int MargLeft, int MargRight, int MargTop, int MargBottom) to append a new entry.

Unless LastPageMaxHeight is set to true, all pages after the last definition will use the properties of the last definition.

Tip: To retrieve a page as meta file use GetPageAsMetafile.

Please see this demo. This interface is used by PageSizeList.

4.6.2.1 Properties

4.6.2.1 A) AsString

Applies to
IWPPageSizeList

Declaration
```
string AsString;
```

Description
Set and retrieve the page sizes (but not the resolution) as string. This can be useful to cache the rectangle sizes.

4.6.2.1 B) Count

Applies to
IWPPageSizeList

Declaration
```
int Count;
```

Description
The count of defined page sizes - readonly.

4.6.2.1 C) LastPageHeight

Applies to
IWPPageSizeList

Declaration
```
int LastPageHeight;
```

Description
The text height of the last page - readonly.

4.6.2.1 D) LastPageMaxHeight

Applies to
IWPPageSizeList

Declaration

```
bool LastPageMaxHeight;
```

Description
If true (default=false) the last page uses a very large height and the width of the previous page.

This way the last page will hold all text which did not fit into the previous pages.

Please use this property with care. It should not be used when the pagelayout mode was selected.

4.6.2.1 E) Resolution

Applies to
IWPPageSizeList

Declaration
```
int Resolution;
```

Description
This is the resolution for the values provided to Add and AddEx.

4.6.2.1 F) Active

Applies to
IWPPageSizeList

Declaration
```
bool Active;
```

Description
Activates and deactivates the defined page sizes. When it is active (and Count>0) the page sizes defined in the document are overridden and the event OnMeasurePage is not been triggered.

4.6.2.2 Methods

Enter topic text here.

4.6.2.2 A) IWPPageSizeList.Add

Applies to
IWPPageSizeList

Declaration
```
procedure Add(Width: Integer; Height: Integer);
```

Description
Adds a page size element. Only the width and the height are defined, the margins are set to 0. By default all values are measured in twips (Resolution=1440).

4.6.2.2 B) IWPPageSizeList.AddEx

Applies to
IWPPageSizeList

Declaration
```
procedure AddEx(Width: Integer; Height: Integer; MargLeft: Integer; MargRight:
Integer; MargTop: Integer; MargBottom: Integer);
```

Description
Adds a page size definition. You can use a nagative value to let a certain value beeing ignored. By default all values are measured in twips (Resolution=1440).

4.6.2.2 C) IWPPageSizeList.Clear

Applies to
IWPPageSizeList

Declaration
```
procedure Clear;
```

Description
Clears all page sizes.

4.6.2.2 D) IWPPageSizeList.Delete

Applies to
IWPPageSizeList

Declaration
```
procedure Delete(Index: Integer);
```

Description
Delete a page size element with a given number. The first element has the index 0.

4.6.3 IWPPrintParameter

Optimize printing (duplex, tray selection)

Description
Change Printing Options. You can use property PageList to print a list of pages, or print the odd or even pages only (PageSides).

You can use IWPMemo.TextCommandStr to get a list of paper tray names: Command ID 47 to be used with FirstPagePaperSource and AllPagePaperSource.

4.6.3.1 Properties

4.6.3.1 A) AllPagePaperSource

Applies to
IWPPrintParameter

Declaration
```
int AllPagePaperSource;
```

Description
The paper tray ID used for all pages.

The selected ID will be saved to RTF but only loaded from RTF when the format string "-readprintpar" was used.

4.6.3.1 B) DontUpdateDEVMode

Applies to
IWPPrintParameter

Declaration
```
bool DontUpdateDEVMode;
```

Description
If true TextDynamic will not try to modify printer settings.

4.6.3.1 C) DuplexMode

Applies to
IWPPrintParameter

Declaration
```
int DuplexMode;
```

Description
0=DontChangeDuplex
1=None
2=orizontal
3=Vertical

4.6.3.1 D) FirstPagePaperSource

Applies to
IWPPrintParameter

Declaration
```
int FirstPagePaperSource;
```

Description
The paper tray ID used for first page only.

The selected ID will be saved to RTF but only loaded from RTF when the format string "-readprintpar" was used.

4.6.3.1 E) Options

Applies to
IWPPrintParameter

Declaration
```
int Options;
```

Description
Option bits for printing and user interface:
bit 1: IgnoreBorders
bit 2: wpIgnoreShading
bit 3: IgnoreText

bit 4: IgnoreGraphics
bit 5: UsePrintPageNumber
bit 6: DoNotChangePrinterDefaults
bit 7 :DontPrintWatermark
bit 8: DontReadPapernamesFromPrinter
bit 9: AlwaysHideFieldmarkers
bit 10:DontAllowSelectionPrinting

4.6.3.1 F) PageList

Applies to
IWPPrintParameter

Declaration
```
string PageList;
```

Description
Page list as string, i.e "1-5,8,10"

4.6.3.1 G) PageSides

Applies to
IWPPrintParameter

Declaration
```
int PageSides;
```

Description
0=print all, 1=print even, 2=print odd pages only

4.6.3.1 H) Title

Applies to
IWPPrintParameter

Declaration
```
string Title;
```

Description
Title for the printer job.

4.6.3.2 Methods

4.6.3.2 A) IWPPrintParameter.SetExtraProp

Applies to
IWPPrintParameter

Declaration
```
procedure SetExtraProp(ID: Integer; Value: Integer);
```

4.6.4 IWPMeasurePageParam

This interface is used by the OnMeasurePage event.

Description
You can use this event to change the page size of any of the pages in the text buffer. The size will be only virtually changed. You need to assign **Changed = true**, otherwise the modified properties are not used

Properties	Properties
Changed	**MarginLeft**
ColCount	**MarginRight**
Height	**MarginTop**
MarginBottom	**PageNr**
	Width

4.6.4.1 Properties

4.6.4.1 A) Changed

Applies to
IWPMeasurePageParam

Declaration
```
bool Changed;
```

Description
Assign true to activated the changes.

4.6.4.1 B) ColCount

Applies to
IWPMeasurePageParam

Declaration
```
int ColCount;
```

Description
Reserved.

4.6.4.1 C) Height

Applies to
IWPMeasurePageParam

Declaration
```
int Height;
```

Description
Height of the page in twips (inch/1440)

4.6.4.1 D) MarginBottom

Applies to
IWPMeasurePageParam

Declaration
```
int MarginBottom;
```

Description
Bottom margin in twips (inch/1440)

4.6.4.1 E) MarginLeft

Applies to
IWPMeasurePageParam

Declaration
```
int MarginLeft;
```

Description
Left margin in twips (inch/1440)

4.6.4.1 F) MarginRight

Applies to
IWPMeasurePageParam

Declaration
```
int MarginRight;
```

Description
Right margin in twips (inch/1440)

4.6.4.1 G) MarginTop

Applies to
IWPMeasurePageParam

Declaration
```
int MarginTop;
```

Description
Top margin in twips (inch/1440)

4.6.4.1 H) PageNr

Applies to
IWPMeasurePageParam

Declaration
```
int PageNr;
```

Description
The page number of the page.

4.6.4.1 I) Width

Applies to
IWPMeasurePageParam

Declaration
```
int Width;
```

Description
The width of the page in twips (inch/1440)

4.8 Mailmerge

"Mail merge" makes it possible to update fields which were embedded in the text with external data. Due to it versatility it is one of the most popular features offered by TextDynamic.

Example:
This is the name field: «NAME»

After first call to MergeText() we read:
This is the name field: «Julian Ziersch»

After second call to MergeText() we read:
This is the name field: «WPCubed GmbH»

(The field markers « and » can be made invisible.)

To use the mail merge feature you only need to

a) insert fields
Method: TextCursor.InputField

It is also possible to convert tokens into fields (TextCursor.FieldsFromTokens), also Syntax Highlighting is possible. An alternative is the "Token To Template Conversion".

b) create callback which fills in data
Event: OnFieldGetText
c) start the merge process
Method: Memo.MergeText

You can use Memo.MergeText each time the data which should be inserted has been updated. So it is easy to create a *viewing* form for database records. The previous contents of the field will be updated. The field is not destroyed by the merge process, such as it is in the usual search&replace approach.

With TextDynamic you can also read out the text inside of a field when the event OnFieldGetText is triggered. So you can also create a database *input* form.

If you want to do mass mailing you can execute MergeText and **append the merged** text to a different, internal text buffer. Please see our create list and label printing demo which show how to use this powerful feature.

Please see the example for InputCell if you need to dynamically create a table with inserted merge fields.

Technical note:

Like hyperlinks merge fields use paired objects. One object marks the start of the field, the other the end. The text within is the "embedded text". You can customize the component to show or hide the fieldmarkers (SpecialTextAttr(wpInsertpoints). Hidden). It is also possible to display a single object only which shows the name of the field. (property IWPMemo.ShowFields)

Category
Mailmerge

4.8.1 Create Merge Fields

There are several ways to create a merge fields in a document:

a) Use the API to create a field:

To insert a field use the method InputField. It expects the name of the field and the initial text. The boolean parameter controls if the cursor should be placed inside (true) or after the new field (false). Using the *true* you can insert multiple paragraphs or an image inside of the field.

```
VB:
TextCursor.InputField "NAME", "Name-Field", False

C#:
TextCursor.InputField("NAME","Name-Field",false);
```

b) Load RTF code which contains fields:

TextDynamic uses standard RTF tags to load and save fields so merge fields inserted in Word are usually available in TextDynamic as well (without the extended attributes).

c) Load RTF text which contains tokens and let the tokens convert into fields:

You can inserted the tags using escape text, i.e. <name> and use either
- the method IWPTextCursor.FieldsFromTokens(StartText,EndText,FieldPreText) (=ReplaceTokens)
- or TextCommandStr(17) "Token To Template Conversion". The latter can ignore start codes which are used inside the text.

You can use TextCommandStr(16) activate syntax highlighting to display the tokens as bold red text.

Note: Fields are displayed as « and ». **Memo.ShowFields** = true to display field names instead and `WPDLLInt1.SpecialTextAttr(SpecialTextSel.wpInsertpoints).Hidden = true` to hide fields.

Note: It is possible to **delete** certain fields with Memo.TextCommandStr.

It is now also possible to the the Memo.Save and Load methods to read and write the data of a certain field. Please see Load and Save Category - Formatstrings.

4.8.2 Insert Data

After the method MergeText was called the event OnFieldGetText will be triggered for each field.

Please use this event to prvide the field data.

This event receives the following parameters:

Editor, the number of the editor, usually 1
FieldName, the fieldname,
Contents, the interface IWPFieldContents to change the contents.

The interface **IWPFieldContents** has many options, but usually you will only need to modify the property StringValue:

VB:

```
Private Sub WPDLLInt1_OnFieldGetText(
  ByVal Editor As Long,
  ByVal FieldName As String,
  ByVal Contents As WPToolsInt.IWPFieldContents)

  If FieldName="NAME" Then
     Contents.StringValue = 'Julian Ziersch'
  End If

End Sub
```

C#

```
Private void wpdllInt1_OnFieldGetText(
      Object Sender,
      int Editor,
      String FieldName,
      WPDynamic.IWPFieldContents Contents)
      {
            If (FieldName.Equals("NAME"))
                  Contents.StringValue = "Julian Ziersch";
      }
```

In the event OnFieldGetText you can access a database, calculate text or use fixed strings. The inserted text can be standard non formatted text, and it can be also RTF or HTML text.

Insert an image

You can insert a picture. If you use the OCX you can use any IPicture reference.

When using .NET convert an "Image" using the Image2Picture class:

```
Contents.LoadPicture(
  new WPDynamic.Image2Picture(
  pictureBox1.Image ),0,0 );
```

Change font attributes

The interface Contents.FieldAttr allows it to change the attributes of the inserted text, i.e. you can use it, for example, to make negative numbers red.

To start the mail merge use the method **MergeText**

4.8.2.1 Use event MS Access

IWPFieldContents is passed as untyped "object", so please create a variable of the type IWPFieldContents and assign it:

```
Private Sub WPDLLInt1_OnFieldGetText(
            ByVal Editor As Long,
            ByVal FieldName As String,
            ByVal Contents As Object)
      Dim theContents As WPTDynInt.IWPFieldContents
      Set theContents = Contents
      If FieldName = "name" Then
            theContents.stringvalue = "Julian Ziersch"
End If
```

4.8.2.2 If Interface cannot be used

If the parameter *Contents* is not accessible as interface IWPFieldContents you can also load the text using Memo.SelText. The TextCursor methods can also be used inside this event and the field text is selected while the event is active.

In Visual FoxPRO this can be helpful:
ThisContents=GETINTERFACE(Contents, "IWPFieldContents", "WPTDynInt.ocx")

Tip: In case You cannot use the event, You can can enumerate the fields using TextCursor.CPMoveNextObject.

4.8.3 Highlight/HideFields

To highlight the fields in the text you can use this code:

VB

```
WPDLLInt1.SpecialTextAttr(wpInsertpoints).Hidden = True
```

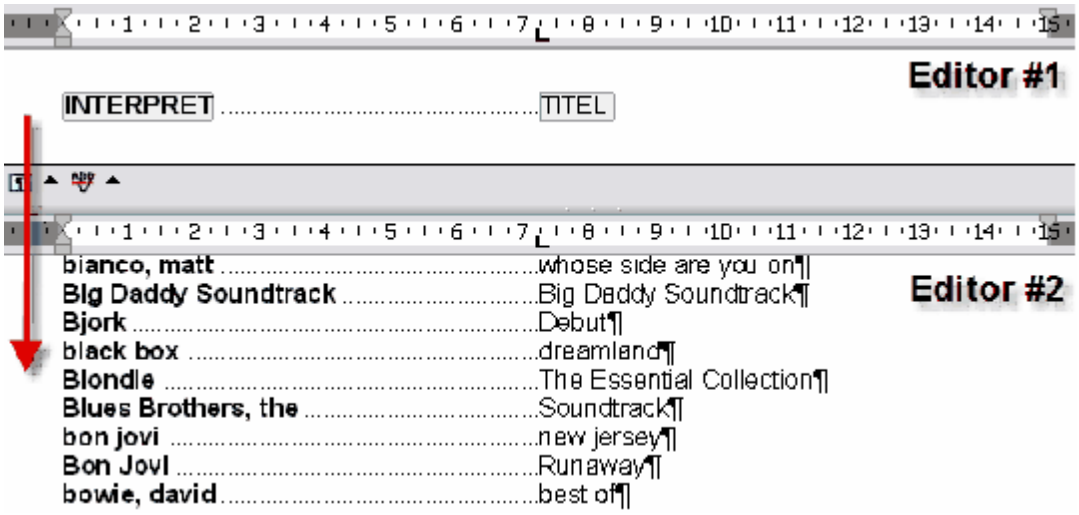
C#

```
WPDLLInt1.SpecialTextAttr(SpecialTextSel.wpInsertpoints).Hidden = true;
```

4.8.4 Append to Editor #2

You can use both editors to create a list or multiple copies of the form letter in editor 1

Use method SetEditorMode to initialize the "double editor" mode.



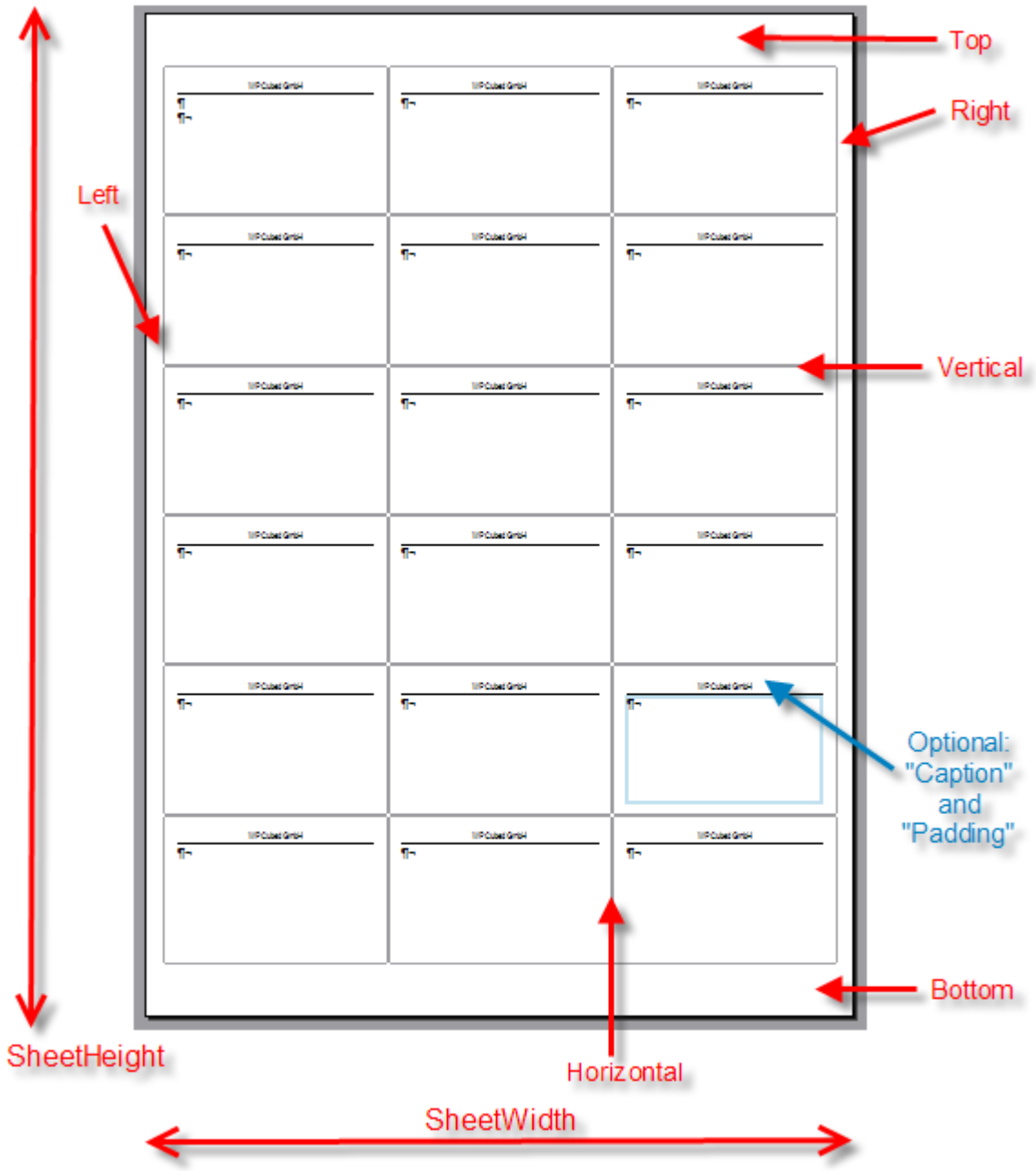
To create the list execute Memo.MergeText for each record and append the mrged text to the editor #2 using Memo2.**AppendOtherText**(0);

In case you want to append to an invisible alternative text buffer use the API Memo. **RTFDataAppendTo** like it is showed in the label demo.

4.8.5 Create Mailing Labels

Now TextDynamic has an exciting new feature. It can display the current document as labels on a label sheet. The label sheet can be configured using the interface LabelDef. You can set the width and height of the sheet, the count of columns and rows, the margins on each side of the sheet and the margin between labels, horizontally and vertically.

The properties of the IWPLabelDef interface as graphic:



Floating point properties such as "SheetHeight" and "SheetWidth" are measured in **centimeter**, the can be also set as **inch** if the property UnitIsInch = true.

The property "Caption" makes it easy to provide a return address. When this string is not empty 1/2 cm will be reserved on the label for the text and a line will be drawn. You can get a similar effect with standard, repeated text on the label text, but since the caption is not part of the text the user cannot delete it. This makes it easier to edit the created labels in the preview before they are printed. (Yes, you can edit in the "Preview"!)

It is also possible to start with a certain label on the first page - in case the label sheet is not empty (StartNr).

The property **AsText** makes it easy to save the label definition as a string together with its name - you can use it to save a set of predefined labels in a simple string list.

To display the label sheet you only have to set the property **Active = true**. This will override the page setup in the editor and the OnMeasurePage event. Since the label definition is part of a "RTFData" element (see demo Simulated MDI) make sure you change it after a call to RTFDataSelect. You can simply use RTFDataSelect("@@FIRST@@") when you are finished with label printing to return to the mailing template.

4.8.5.1 C# Code

This C# code uses a loop instead of a database and the event OnFieldGetText is not defined.

```
// Initialize
   IWPMemo memo;
   IWPTextCursor cursor;
   memo = wpdllInt1.Memo;
   cursor = memo.TextCursor;
   memo.RTFDataSelect("@@FIRST@@"); // undoes memo.RTFDataSelect("LABELS")
   memo.Clear(false, true);
   IWPCharAttr CurrAttr = memo.CurrAttr;

// Create mailing fields
   CurrAttr.Clear();
   CurrAttr.SetFontSize(10);
   cursor.InputField("NAME","Name",false);
   cursor.InputString("\r",0);
   cursor.InputField("ADR1","Adr1",false);
   cursor.InputString("\r",0);
   cursor.InputField("ADR2","Adr2",false);
   cursor.InputString("\r\r",0);
   CurrAttr.IncludeStyles(1); // bold
   cursor.InputField("ZIP","00000",false);
   cursor.InputString(" ",0);
   cursor.InputField("CITY","City",false);

// Now merge text text and copy it to a
// different text element using RTFDataAppendTo
// In this example we just do a loop. Usually
// you will scroll through the database in this loop

// Please Note: first merge, then append!

   memo.RTFDataDelete("LABELS"); // Clear it
   for (int i=0; i<30; i++)
   {
      memo.MergeText("");
      memo.RTFDataAppendTo("LABELS", 1);
   }

// Select the label sheet
// Do it before you configure the label!
   memo.RTFDataSelect("LABELS");

// Switch off the field markers
   memo.SpecialTextAttr(SpecialTextSel.wpInsertpoints).Hidden = true;

// Set up the label sheet
   IWPLabelDef label;
   label = memo.LabelDef;
   label.Caption = "WPCubed GmbH * St. Ingbert Str. 30 * 8151 Munich";
   label.ColumnCount = 2;
```

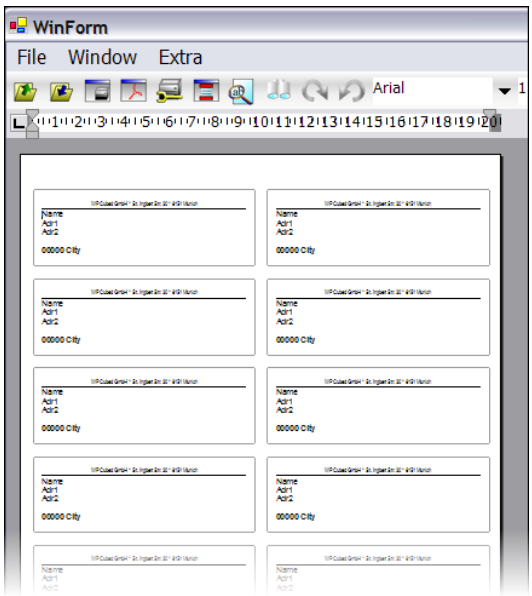
```
    label.RowCount = 7;
    label.Horizontal = 0.5F; // 5 mm between labels
        label.Vertical = 0.5F; // 5 mm between labels
// and display it
        label.Active = true;
```

4.8.5.2 VB Code

```
        ' The interfaces we need
        Dim memo As WPDynamic.IWPMemo = Me.WpdllInt1.Memo
        Dim cursor As WPDynamic.IWPTextCursor = memo.TextCursor

        ' Create the mailing template
        memo.RTFDataSelect "@@FIRST@@" ' undoes memo.RTFDataSelect("LABELS")
        memo.Clear(False, True)
        memo.CurrAttr.Clear()
        memo.CurrAttr.SetFontSize(10.0!)
        cursor.InputField("NAME", "Name", False)
        cursor.InputString(ChrW(13), 0)
        cursor.InputField("ADR1", "Adr1", False)
        cursor.InputString(ChrW(13), 0)
        cursor.InputField("ADR2", "Adr2", False)
        cursor.InputString(ChrW(13) & ChrW(13), 0)
        memo.CurrAttr.IncludeStyles(1)
        cursor.InputField("ZIP", "00000", False)
        cursor.InputString(" ", 0)
        cursor.InputField("CITY", "City", False)

        ' Simulate a mail merge - do a loop in this example
        ' Please Note: first merge, then append!
        Dim i As Integer
        For i = 1 To 30
            memo.MergeText("")
            memo.RTFDataAppendTo("LABELS", 1)
        Next

        ' Select the resulting text
        memo.RTFDataSelect("LABELS")

        ' Hide the field markers
        memo.SpecialTextAttr( _
            WPDynamic.SpecialTextSel.wpInsertpoints).Hidden = True

        ' Define the label properties
        Dim label As WPDynamic.IWPLabelDef = memo.LabelDef
        label.Caption = "WPCubed GmbH * St. Ingbert Str. 30 * 8151 Munich"
        label.ColumnCount = 2
        label.RowCount = 7
        label.Horizontal = 0.5!
        label.Vertical = 0.5!

        ' ... and activate it
        label.Active = True
```

4.8.5.3 Resulting Label Sheet

The resulting label sheet is displayed in the editor and can be printed and edited (!)



4.8.6 interface IWPFieldContents

Work with mail merge fields

Description
This interface is passed as parameter Contents to the event OnFieldGetText. It is used to provide the field data when doing mail merge or reporting.

To update the contents of the field simply assign a string to property StringValue - this also works for formatted text.

You can use the method ContinueOptions to change the way the text is inserted, for example to ignore some of the loaded attributes.

It is also possible to create or update an image which is placed inside the field. To make this as easy as possible there are two functions LoadImage and LoadPicture. Both methods will not simply replace the contents of the field with a new image but reuse an existing image container object.

To create a hyperlink inside the field use InputHyperlink. You can also load a file with formatted text using LoadText

To create a table with data inside the field execute AddTable. The event OnCreateNewCell can be used to format and fill each new cell.

If you know that you do not need the field after the merge process you can execute DeleteField. The field markers, not the contents will be removed.

An interface to access the current field is provided as property FieldObject. You can use this interface to read other properties of the merge field.

The reporting engine will also trigger the event OnFieldGetText to evaluate fields which were used in band conditions such as ?fieldname=null and ?fieldname#null. In this

case the property IsMergefield =false. In this case use ExecStrCommand to modify the state property.

Properties	Methods
Description	**AddTable**
FieldCommand	**ContinueOptions**
FieldName	CurrentBand
FloatValue	CurrentGroup
Format	**DeleteField**
IsMergefield	**EmbeddedObject**
StringValue	**ExecStrCommand**
Title	**FieldAttr**
	FieldObject
	InputHyperlink
	LoadImage
	LoadPicture
	LoadText
	SetValue

4.8.6.1 Properties

4.8.6.1 A) Description

Applies to
IWPFieldContents

Declaration
```
string Description;
```

Description

Only used during report creation to red predefined value.

4.8.6.1 B) FieldCommand

Applies to
IWPFieldContents

Declaration
```
string FieldCommand;
```

Description
The property gives access to the "command" property of the field.

4.8.6.1 C) FieldName

Applies to
IWPFieldContents

Declaration

```
string FieldName;
```

Description
The property gives access to the name of the field.

4.8.6.1 D) FloatValue

Applies to
IWPFieldContents

Declaration
```
double FloatValue;
```

4.8.6.1 E) Format

Applies to
IWPFieldContents

Declaration
```
string Format;
```

Description

```
Only used during report creation to red predefined value.
```

4.8.6.1 F) IsMergefield

Applies to
IWPFieldContents

Declaration
```
bool IsMergefield;
```

Description
The reporting engine will also trigger the event OnFieldGetText to evaluate fields which were used in band conditions such as ?fieldname=null and ?fieldname#null. In this case the property IsMergefield =false;

4.8.6.1 G) StringValue

Applies to
IWPFieldContents

Declaration
```
string StringValue;
```

Description
This property is used to assign a string to this value to replace the contents of the field. You can also read the field to retrieve the current data. An instance of the interface IWPFieldContents is passed to the event OnFieldGetText after the method MergeText was called.

You can also assign a string in RTF or WPT format. HTML will be auto detected if the text starts with <html>.

Example:

```
StringValue = "<html><b>Super</b> mail merge</html>"
```
This code will insert the text: **Super** mail merge

If you need to create a hyperlink use the method InputHyperlink und set the link text using this property.

Note

If you cannot use the IWPFieldContents interface because of a limitation of the developing language, you can insert text using the method LoadFromString. The input methods of the IWPTextCursor can also be used, only the cursor position or selection may not be changed. In MS-Access™ please define a global variable to hold a IWPFieldContents interface reference. In the event OnFieldGetText assign the "Contents" parameter to that variable.

Category
Mailmerge

4.8.6.1 H) Title

Applies to
IWPFieldContents

Declaration
```
string Title;
```

Description

```
Only used during report creation to red predefined value.
```

4.8.6.2 Methods

4.8.6.2 A) IWPFieldContents.AddTable

Applies to
IWPFieldContents

Declaration
```
procedure AddTable(ColCount: Integer; RowCount: Integer; Border: WordBool;
EventParam: Integer; CreateHeader: WordBool; CreateFooter: WordBool);
```

Description
This is a powerful and versatile method. It creates a table inside the merge field inside the mailmerge process. The callback OnCreateNewCell is triggered if EventParam was passed with a value != 0. So you can add the data and properties is a very efficient way.

Note

If you do not need to merge the template again we recommend to use DeleteFields. This makes sure there are no extra empty lines before and after the table which are required to host the field markers.

Parameters	
ColCount	The count of columns which should be created
RowCount	The count of rows which should be created (not including header/footer rows). You can abort the creation in event OnCreateNewCell by changing the value of the boolen variable AbortAtRowEnd.

Border	Create a broder around cells. You can also set border attributes in event OnCreateNewCell.
EventParam	Parameter passed through to OnCreateNewCell. This event is not called if this parameter is 0.
CreateHeader	If true an extra header row with rownr=-1 will be created. The formatting routine is able to repeat header rows at the start of a page.
CreateFooter	If true an extra footer row with rownr=-2 will be created. The formatting routine is able to repeat footer rows at the start of a page. (This special feature is not supported by MSWord)

Category
Callback Functions
Table Support

4.8.6.2 B) IWPFieldContents.ContinueOptions

Applies to
IWPFieldContents

Declaration
```
void ContinueOptions(int OptionBits);
```

Description
This method is used to **add** flags which change the way the inserted text (or the insertion itself) is performed for this field.

You can use the following bit values:
1: The paragraph attributes of the inserted text will be overridden with the attributes of the paragraph the field is located within.
2: The character attributes of the inserted text will be overridden with the character attributes of the field object.
4: Do not load font information from the inserted RTF text.
8: Do not load font size information from the inserted RTF text.
16: Do not load font style information (bold, italic, underline) from the inserted RTF text.
32: Do not load tabs - use current tabs.
64: Do not load paragraph styles - use current style.

Usually the loaded text will use the current paragraph attributes for its first paragraph.

The following bits change this behaviour:
128: Overwrite current paragraph attributes with attributes from first loaded paragraph.
256: If merge field is at start of paragraph overwrite current paragraph attributes with attributes from the first new paragraph.

This bits refine the copying of the properties:
512: Also use the borders of loaded text (overwrites table cell borders)
1024: Do not use the loaded tab positions
2048: Do not use the loaded paragraph style

When inserting text the fields (start/end marker) stay intact.
Only if this bit is set, the field will be deleted:
4096: Delete This Field

Note

This method will add flags but will not remove flags which have been set before. The flags are only used for the current field and then set to 0.

Category
Mailmerge

4.8.6.2 C) IWPFieldContents.CurrentBand

Applies to
IWPFieldContents

Declaration
```
function CurrentBand: IWPReportBand;
```

4.8.6.2 D) IWPFieldContents.CurrentGroup

Applies to
IWPFieldContents

Declaration
```
function CurrentGroup: IWPReportBand;
```

4.8.6.2 E) IWPFieldContents.DeleteField

Applies to
IWPFieldContents

Declaration
```
procedure DeleteField;
```

Description
Use this method if you need to delete the field markers. If you delete the fields you cannot use the template for merging again. But if the inserted text contained fields those can be merged in a second round. If you insert a table (AddTable) DeleteField is required to avoid otherwise blank paragraphs which host the field markers.

Note: The markers are not deleted when you call this function but later when the field is processed and the data specified using StringValue is inserted.

4.8.6.2 F) IWPFieldContents.EmbeddedObject

Applies to
IWPFieldContents

Declaration
```
function EmbeddedObject: IWPTextObj;
```

Description
This reference allows it to manipulate the first object, probably an image, inside the merge field. If no object is within the field, EmbeddedObject will be null.

4.8.6.2 G) IWPFieldContents.ExecStrCommand

Applies to
IWPFieldContents

Declaration
```
procedure ExecStrCommand(CommandID: Integer; param: Integer; const StrParam:
WideString);
```

CommandID = 1:

Set the state of the field using the parameter 'param' inside the event OnFieldGetText. The State is used for band conditions of the integrated reporting.

0: Default value

1: Field is not known (default if checking field and band conditions, i.e. ?field=null)

2: Field is defined and NULL

3: Field is defined and ZERO

4: Field is defined (set this value unless it is null or zero)

Example - Simulate certain conditions:

```
private void wpdllInt2_OnFieldGetText(object Sender,

    int Editor, string FieldName, IWPFieldContents Contents)
        {
            // test the conditions
            if (!Contents.IsMergefield)
            {
                if (FieldName == "ISNULL")
                    Contents.ExecStrCommand(1, 2, "");
                else if (FieldName == "ISZERO")
                    Contents.ExecStrCommand(1, 3, "");
                else if (FieldName == "ISUNDEF")
                    Contents.ExecStrCommand(1, 1, "");
                else
                {
                    Contents.ExecStrCommand(1, 4, ""); // Known
                    Contents.StringValue = "123";
                }
            } else
            ....
        }
```

This conditions can be used inside band tokens like in this simplified sample (See Token to Template Conversion):

```
<<#GROUP1 "Invoice">> -- the table will be given the display name
"invoice", the logical names remains "GROUP1". After the optional name a
condition is expected.¶
<<:HEADER/>> -- this header will be displayed at the start of the group¶
Group Header. Fields are possible: <<fieldname +spc/>>¶
<<:DATA ?ISNULL#null/>>  -- this is the data row.¶
Some data here - <<rnd>> - under condition a field is not null¶
<<:DATA ?ISNULL=null/>>  -- this is the data row.¶
Some data here - <<rnd>> - under condition a field is null or unknown. ¶
<<:FOOTER/>> -- this footer will be displayed at the end of the group¶
Group Footer. Fields are possible: <<fieldname/>>¶
<<#/GROUP1>> -- close the group. ¶
```

Other CommandIDs:

This method can be extended for custom manipulations of the inserted text - if you have a wish please let us know.

4.8.6.2 H) IWPFieldContents.FieldAttr

Applies to
IWPFieldContents

Declaration
```
function FieldAttr: IWPAttrInterface;
```

Description
This interface allows it to change the attributes of the inserted text. You can use it, for example, to make negative numbers red.

Category
Character Attributes

4.8.6.2 I) IWPFieldContents.FieldObject

Applies to
IWPFieldContents

Declaration
```
function FieldObject: IWPTextObj;
```

Description
This interface provides access to the field marker itself. You can use this interface to change the name and command properties. To delete a field use DeleteFields.

4.8.6.2 J) IWPFieldContents.InputHyperlink

Applies to
IWPFieldContents

Declaration
```
procedure InputHyperlink(const URL: WideString; const LinkName: WideString);
```

Description
Use this method to create a hyperlink within the field. The link text has to be set using property StringValue. This makes it possible to place formatted text or images inside the link.
The first parameter is the URL, the second is reserved (link caption).

Category
Hyperlinks and Bookmarks

4.8.6.2 K) IWPFieldContents.LoadImage

Applies to
IWPFieldContents

Declaration
```
procedure LoadImage(const filename: WideString; w: Integer; h: Integer);
```

Description
You can load an image into this field. If the field alreaday contains an image object the existing object will be reused.

4.8.6.2 L) IWPFieldContents.LoadPicture

Applies to
IWPFieldContents

Declaration
```
procedure LoadPicture(const Picture: IUnknown; w: Integer; h: Integer);
```

Description
You can insert a picture. If you use the OCX you can use any IPicture reference.
When using .NET convert an "Image" using the Image2Picture class:

```
Contents.LoadPicture(new WPDynamic.Image2Picture(
  pictureBox1.Image ),0,0 );
```

4.8.6.2 M) IWPFieldContents.LoadText

Applies to
IWPFieldContents

Declaration
```
procedure LoadText(const filename: WideString);
```

Description
This method can be used to load a file which will be inserted.

Category
Load and Save

4.8.6.2 N) IWPFieldContents.SetValue

Applies to
IWPFieldContents

Declaration
```
procedure SetValue(Value: OleVariant);
```

4.9 Reporting

TextDynamic contains an optional reporting feature which is controlled by the interface IWPReport.

This reporting can be understood as a "super charged version of mail merge": You can not only update fields with data, you can also use bands for conditional text, to create headers and footers and, most important, loop certain parts of the template until all data has been printed, i.e. to create a list.

Thus internal reporting features can help you to do
- mass mailing
- mass e-mail
- create directories and lists
- create and print invoices
- create contracts which consist of different text blocks

The output can be saved as MSG (=MIME E-Mail Format), RTF, HTML and, optionally, PDF.

The integrated reporting engine can

a) use a template which is regular text with <<tokens>>.
--> Token to Template Conversion

b) use a template in a proprietary format.

c) create and use a DB-Description (XML) which is the basis to create and maintain the template (b)
--> DB-Description and Template architecture

Introduction Project:
VS2008 Reporting Demo

4.9.1 Introduction

In contrast to the the plenty reporting applications and tools available already, our reporting engine is based on a word processor. This means the reporting template is just a text document, so is the output.

This makes the output completely editable after the creation. It is exceptionally useful to edit the output after the creation but prior to the print to quickly check the accuracy of the text, for example if names are written correctly.

Using the new "token to template conversion" You can use a document as input, simply add a few tags (<<..>>) and you can create an invoice or a list at once. This makes it very easy to convert existing reports into reporting templates. It is not required to explore the more advanced reporting features of TextDynamic to do so, just add 2 evens and you are done. One event is responsible to select the data sets (build queries) and advance the dataset cursor, the other event is responsible to retrieve the field data. Usually a few lines of code will do.

Only this two events needs to be used:
OnFieldGetText
OnReportState

RTF2PDF / TextDynamic Server now also comes as reporting edition which can work with the same templates.

4.9.1.1 Comparison to "usual" reporting

1) End user can modify templates

Many reporting tools are very complicated to use or do not offer end user modifications at all.

a) With TextDynamic it is possible to load a regular document with added tags for fields (i.e. **<<name>>**) and bands (i.e. **<<#ORDERS>> .... <<#/ORDERS>>**) and TextDynamic can convert such a document into a reporting template (=token to template conversion). Syntax highlighting is possible while editing the reporting template in text form.

b) If the database structure is rather complicated you can use the alternative template architecture. Here the report logic has been split up into two parts (pre template and template architecture).

One part is created and maintained by the developer. It contains the outline of the report with bands which can but do not have to be used, fields which can be used and variables which are calculated while the report is being processed as XML data.

The second part is the report template. It can be edited by the end user (you can of course hide it, too) using the word processor. It is possible to insert fields and variables from the "repository". It is also possible to reset a selected group to its initial state in case it has been messed up. It is saved as document with added proprietary attributes to maintain the bands.

2) No fixed page layouts

Most reporting tools work with page layouts. This means you may place graphics at exact positions on a form and when the report is created it will exactly look as the designed forms.

This approach is often very good, but is some cases you want the text created by the reporter to be edit able. The mentioned approach has a problem here - while it is sometimes possible to create RTF documents with the reporting tool theses are not really "edit able" because the text has been broken up into the tiniest pieces - just single text boxes. Usually the RTF export is optimized to be best viewed in MS Word only.

With the TextDynamic report a text file (RTF or WPT format) is created which allows full editing, including changing of the page size, and other operations which makes the re-pagination of the text necessary.

This means that the reporting feature is optimal if you need to create longer texts, such as contracts. It will be suitable to print lists and invoices. If you need to print forms which look like pre printed paper forms (i.e. TAX forms) you cannot use the TextDynamic reporting. But in this case you will probably not need sophisticated formatted text.

3) Most RTF attributes usable

Usually the reporting tools only support very limited RTF features, sometimes even justified text is a problem, not to speak of images with text wrapping around, tables and more sophisticated tab stops which use fill signs.

Since the TextDynamic reporting uses the same text engine all the powerful word processing features can be used in the created report.

4) No external installation

Many reporting tools require to be installed separately, they can be quite cost intensive and the licensing can be restrictive.

The TextDynamic reporting can be simply enabled by purchasing the add on license key at a very low price.

No additional installation is required. Since for report print and preview, data-merge and loading plus saving the report templates the same code is used which is used by the word processor, the size overhead is extremely small.

4.9.1.2 Example

As Introduction a little example to show what can be done with the TextDynamic reporting feature easily:

Here we feature the token to template conversion.

The template

```
Dear <<CustomerName>>,

this are the items you ordered:

<<#ITEMS>>
<<:HEADER>>
Itemname    Count        Price
<<DATA/>>
<<NAME/>><<COUNT/>>      <<PRICE/>>
<<FOOTER/>>
Total                    <<ORDERTOTAL/>>
<<#/ITEMS>>

We will ship the items on <<ShippingDate>>
```

Can be quickly converted into an order confirmation:

```
Dear Michael Miller,

this are the items you ordered:

Itemname    Count        Price
Mousepad    1            10.00
Optical Mouse 1          30.00
Total                    40.00

We will ship the items on 31.3.2008.
```

4.9.2 Token to Template Conversion

If you use the *token to template conversion* you can edit the reporting template with an editor such as MS Word.

The template is loaded into TextDynamic. In this control the template can be further edited with the additional convenience of syntax highlighting and on demand converted into a "true" reporting template. (The latter uses nested paragraphs to represent groups and special

paragraph types for bands. Fields are not text based but use objects instead.).

Editing is not possible in the non visual RTF2PDF TextDynamic Server - we include an external application which offers the syntax highlighting.

If you used the mail merge feature before and have documents with embedded merge fields those can be easily used a reporting templates as well! It is also possible to convert an existing report into a template.

Start the Syntax Highlighter
Memo.TextCommandStr ID 16 - Syntax Highlighting

Start the Token to Template conversion:

Use Report.ConvertTokens or

Memo.TextCommandStr ID 17 - Token To Template Conversion

Note: The codes <<, >>, : and # can be customized in the above two methods!

Overview:
General Syntax - Fields, Bands, Groups
Parameters for Fields, Parameters for Bands and Groups
Example Template

4.9.2.1 General Syntax - Fields

The token to template conversion uses special character combinations to separate the commands from the text. Fields have to be embedded into the characters **<<** and **>>**.

(This codes <<, >>, : and # can be customized in Memo.TextCommandStr)

Note:

The parser expects that after the << and before the >> characters no whitespace characters are typed. Otherwise the characters will be interpreted as text.

Example for fields:

<<name>>

The name of a field may contain space characters.

Optionally the closing '/' character can be printed:

<<name/>>

Using such fields mail merge documents can be created. Such merge documents only require fields to be filled with data. Using mail merge does not require the reporting extension so we decided to make the syntax highlighting and token to template conversion available in the basis edition.

If you need repeated data rows in a document you need groups and bands. Bands mark certain text to be the header and footer of a document or group while groups are used to loop certain parts of the template as long as data to fill in the fields is available. Groups can also be used to disable certain parts of the template. While bands are not nested, groups can be nested and so consist of a start and end token.

Note: The name of a field will be passed to the event OnFieldGetText while merging the text or creating a report. This event can retrieve the text which should be inserted inside the field.

TextDynamic includes a **syntax highlighter** for XML and also for the syntax described in this chapter - please see Memo.TextCommandStr(16) - Syntax Highlighting

4.9.2.2 Bands

Bands are identified by a colon ('**:**') after the opening << characters.

Example for header bands:

<<:HEADER>>

This starts a group header or, if used outside of any group a document header text. In the latter case the following variations are possible:

HEADER**F** = header on first page only
HEADER**O** = header on odd pages
HEADER**E** = header on even pages
HEADER**_OFF** = this header is disabled.

Please note that bands are not implemented using opening and closing tokens ("i.e. <<..>>... <</...>>") so optionally the closing '/' character can be typed: <<:HEADER/>>

Example for footer bands:

<<:FOOTER>>

This starts a group footer or, if used outside of any group a document footer text. In the latter case the following variations are possible:

FOOTER**F** = footer on first page only
FOOTER**O** = footer on odd pages
FOOTER**E** = footer on even pages
FOOTER**_OFF** = this footer is disabled.

Inside header and footer bands fields, images and text is possible. A header and footer band ends where either a different header or footer band starts or a data band:

<<:DATA/>>

Bands must be the **first non white space** (white space= space or tab characters) in a paragraph. All text after the band token will be ignored and can be used to write comments.

4.9.2.3 Groups

Groups are identified by a double cross ('#') after the opening << characters. While bands <<:.../>> are not nested, groups can be nested and so are embedded into a tag pairs **<<#...>> <<#/....>**.

Unlike header and footer bands the name of the group tags are not fixed. Practically any name can be used if it does not contain spaces. However the opening token must match the closing token. For practical reasons the names should match the database which is referenced inside of the group.

Example:

```
<<#CUSTOMERS>> comment: we list all customers in this group
```

```
        <<Customers.Name/>>
        <<Customers.Address/>>
        Ordered Items:
        <<#ORDERS>> comment: we list all items ordered by the current customer
              <<Orders.Name/>>
        <<#/ORDERS>>
<<#/CUSTOMERS>>
```

Note: The name of a group will be passed to the event OnReportState while creating a report. This event has to decide if a group should be processed (again) or not. It can create a sql query and calculate sub totals.

4.9.2.4 Parameters for Fields

1. Name - may contain spaces! (required)

   If first character is the @ sign, the complete text will be handled as formula. (no other options are possible)

2. Displayname in " " - must be second position !

3. Options.
   +spc - this will add a space after the field if it was NOT empty.
   +spc/ - this will add a space before the field if it was NOT empty
   +nl - to add a new-line if it was not empty
   \f "some text" - will be added after the field if it was NOT empty. Alias: + "..."
   \b "some text" - will be added before the field if it was NOT empty. Alias: + "..."/
   -"some text" - will be printed if it WAS empty
   \r "some text" - will be printed instead of the field

Special attributes
   \remove - the field will be removed
   \setattr - the paragraph attribute will be overwritten by field contents attributes
   \autosetattr - if this is first field in line use the para attribute
   \keepattr - protect the current paragraph attributes
   \loadimage - interpret field value as file name of an image file
   \loadtext - interpret field value as file name of a text file

Supports Word Syntax: \b before \f after. Spaces are allowed after +, -, \b, \f ...

4. Format @"...."
   \@ "%f" - uses this formatting code to format the field (usually floating point numbers)

5. Condition ?...=...
   ?fieldname=*null* - use this field if the specified field was null
   ?fieldname#*null* - use this field if the specified field is not null
   ?fieldname=zero - use this field if the specified field was zero
   ?fieldname#zero - use this field if the specified field is not zero

   Use IWPFieldContents.ExecStrCommand to tell the engine that a field is null or zero.

4.9.2.5 Parameters for Bands and Groups

1. Name - predefined header/footer names or one of the possible group names

2. Displayname name in " " - must be second position

3. Option +-
   +ff - start a new page after the band/group
   -ff - start a new page before the band/group

Only header and footer: ("<<:HEADER>>)
   +afterstretch - header/footer used after a streched band
   +between - use band between records
   -start - don't use header at start
   -end - don't use footer at end

Only Data Bands ("<<:DATA>>)
   -stretch - deactivate stretching (default = on)
   +keep - keep this paragraph together with next
   +newtable - start a new table

4. Condition ?...=...
   The following conditions to use a band are possible:

```
=null  - field must be null
#null  - field must not be null
=zero  - field must be zero
#zero  - field must not be zero
="..." - the field must have a certain value
#"..." - the field must be different to a certain value
```

   Example:

```
<<:data ?gender="W">>¶
Dear Mrs. <<name/>>,¶
<<:data ?gender="M">>¶
Dear Mr. <<name/>>,¶
```

   Use IWPFieldContents.ExecStrCommand to tell the engine that a field is null or zero.

5. Formula - must be last option
   @....

It is used as start condition in a group start and a continue condition if used in a group end token.

Any text in the paragraph after the closing '>>' will be ignored as comment!

4.9.2.6 Example Template

```
<<:HEADERF "my header"/>>  -- any text after the group will be ignored as comment
This is the header on the first page
<<:HEADER "my header"/>> -- HEADERF, HEADERO, HEADERE, HEADER
This is the header on all pages
<<:TEXT>> -- start the regular text

Dear Developer,

This letter documents and demonstrates the reporting feature which is based on a
```

template written in regular formatted text with additional tags:

The "<<#" signs are expected as first non-space signs in a paragraph. Any text after the closing >> signs is ignored and can be used as comment. Fields are inserted between the signs << and >>. It is important that after the '<<' no white space or punctuation is written.

Headers in groups are written using single, not open/closing tags. They cannot be nested anyway. The / before the closing >> signs should be added but is not required.

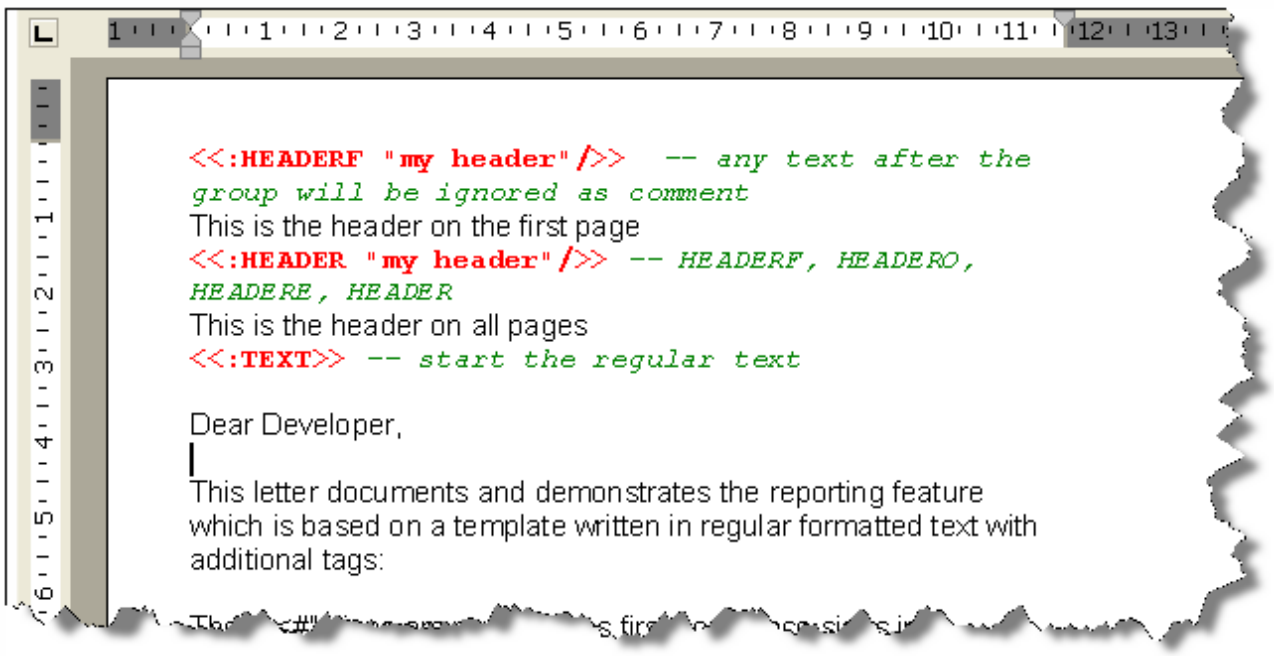
Only groups are nestable. Groups are started using the code <<#. The name after the # sign will be the group name, so a database name can be used. An optional alias can be specified. It is possible to set up a list of possible group names.

Now our list starts:

<<#GROUP1 ?name#null "Invoice">> -- the table will be given the display name "invoice", the logical names remains "GROUP1". After the optional name a condition is expected.
<<:HEADER/>> -- this header will be displayed at the start of the group
Group Header. Fields are possible: <<fieldname +spc/>>
<<:DATA ?"has orders"=null/>> -- this is the data row.
Some data here - under condition that there are no orders. (Note that a field name may used spaces. In this case use " in the field name part as well.)
<<:DATA ?hasorders#null/>> -- this is the data row.
Some data here - under condition there are orders.
<<:FOOTER/>> -- this footer will be displayed at the end of the group
Group Footer. Fields are possible: <<fieldname/>>
<<#/GROUP1>> -- close the group.

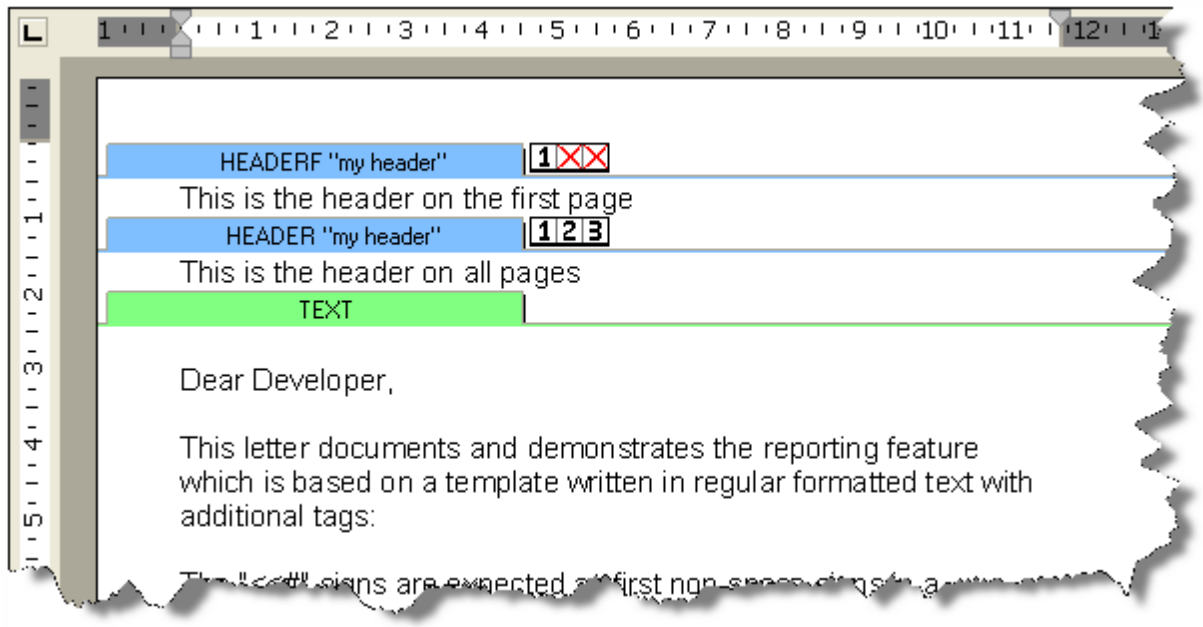
This text comes after the group.

When viewed with activated the **syntax highlighting** the document will look like



Note: The syntax highlighter does not make any modifications to the attributes of the text. All highlighting is done just visual and is updated while the text is edited.

After the conversion to the internal reporting template structure the document looks like this:



4.9.3 Event driven reporting

To completely control the report creation use the events of TextDynamic.

There are three events, one controls the group processing,
one reads the values of fields and one reads out variables.

OnReportState
OnFieldGetText
OnReadFormulaVar

To outline the idea of the events we use some simple C# code:

OnReportState:

This event is triggered before and after a group is started. You use it to move to the next data record or initalize a query. Here we simply use the Count property to abort after 10 rows.

```
private void wpdllInt1_OnReportState(
        object Sender, string Name,
        int State, WPDynamic.IWPReportBand Band,
        ref bool Abort)
{
        if (State==WPDynamic.commands.REP_BeforeProcessGroup)
                                Abort = Band.Count>10;
        else Abort = false;
}
```

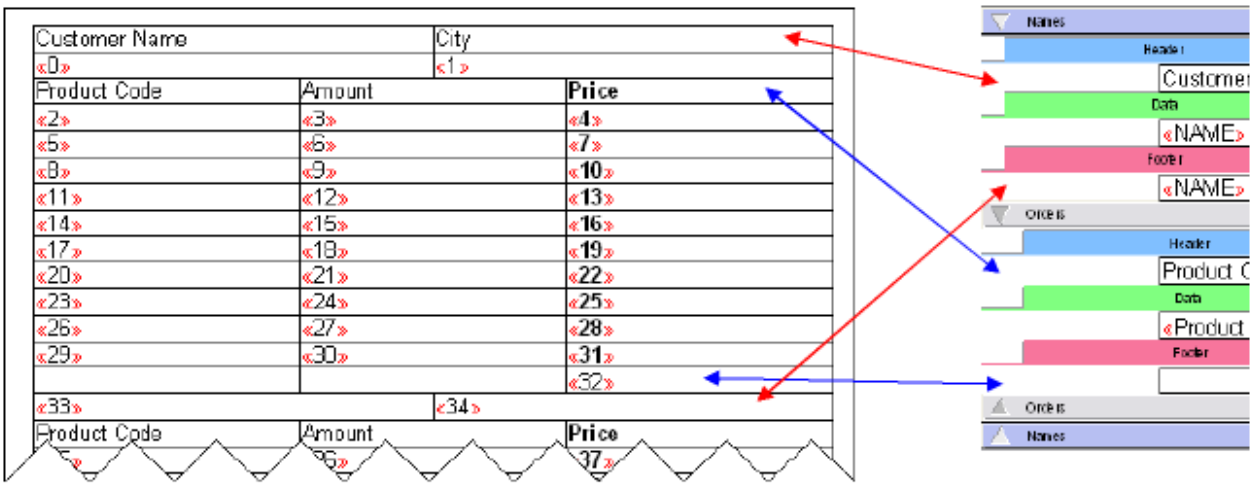
OnFieldGetText:

This event is triggered to fill in field data. It is the same as the one for the regular mail merge.

Here we simply print an incremented number.

```
static int a;
private void wpdllInt1_OnFieldGetText(object Sender, int Editor,
        string FieldName, WPDynamic.IWPFieldContents Contents)
{
        Contents.StringValue = Convert.ToString(a++);
}
```

The image shows the result



OnReadFormulaVar

It is triggered to read the value of a variable which is used in a formula. Please don't mix up with group variables used by the reporting engine, those group variables contain the formulas which itself contain the variables which trigger this event. In fact any unknown name inside a formula will trigger the event OnReadFormulaVar.

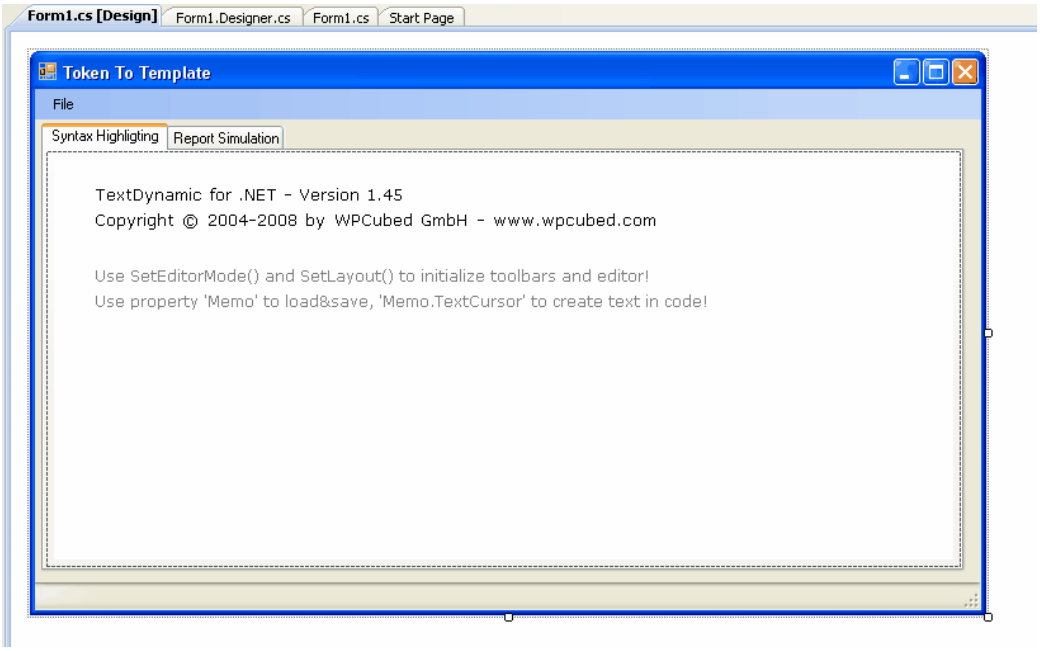
4.9.4 VS2008 Reporting Demo

In this chapter we develop a simple program which can be used to explore the possibilities of the TextDynamic reporting.

4.9.4.1 Basis

Our demo uses a tab sheet with two pages.

One page #1 (`wpdllInt1`) we implement a regular word processor while on page #2 the editor responsible for reporting is placed (`wpdllInt2`).



We initialize both editors in the constructor:

```
using WPDynamic;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            // Initialize the Editor #1
-----------------------------------------
            wpdllInt1.EditorStart("your_name", "your_key");
            // Load the PCC file (no password)
            wpdllInt1.SetLayout("buttons.pcc", "");
            // Initialize the editor
            wpdllInt1.SetEditorMode(
                // as single editor (no splitscreen)
            EditorMode.wpmodSingleEditor,
                // with 16x16 toolbar (for 24x24 use wpmodexToolbarLG)
            EditorXMode.wpmodexToolbar |
            EditorXMode.wpmodexTables,
                // Select the GUI elements ruler, scrollbars and
                // the lower left panel to change zooming and layout
            EditorGUI.wpguiRuler |
            EditorGUI.wpguiHorzScrollBar |
            EditorGUI.wpguiVertScrollBar,
                // We have no second editor window, so use wpguiDontSet
            EditorGUI.wpguiDontSet);

            // Set Default Font
            wpdllInt1.Memo.TextCommandStr(18, 11, "Verdana");

            // Set PageSize (use smaller margins, 720 twips)
            wpdllInt1.Memo.PageSize.SetPageWH(-1, -1, 720, 720, 720, 720);
```

```
            // This should be also used for new documents
            wpdllInt1.Memo.TextCommandStr(19, 0, "");

            // Activate the Syntax Higlighting for fields and bands
            wpdllInt1.Memo.TextCommandStr(16, 4, "");

            // Initialize the Editor #2
----------------------------------------
            wpdllInt2.EditorStart("your_name", "your_key");
            // Load the PCC file (no password)
            wpdllInt2.SetLayout("buttons.pcc", "");
            // A double editor WITH reporting and PDF!
            wpdllInt2.SetEditorMode(
            EditorMode.wpmodDoubleEditor,

            EditorXMode.wpmodexTables |
            EditorXMode.wpmodexPDFExport |
            EditorXMode.wpmodexReporting,

            EditorGUI.wpguiHorzScrollBar |
            EditorGUI.wpguiVertScrollBar
            ,
                // We have no second editor window, so use wpguiDontSet
            EditorGUI.wpguiPanelH2 |
            EditorGUI.wpguiHorzScrollBar |
            EditorGUI.wpguiVertScrollBar |
            EditorGUI.wpguiVertRuler

            );

            // The upper editor should be readonly since we update it from
            // the template
            wpdllInt2.Memo.Readonly = true;
            wpdllInt2.Memo.AutoZoom = AutoZoom.wpAutoZoomOff;

            // Set X offset to 360 twips
            wpdllInt2.Memo.SetIProp(5, 360);

            wpdllInt2.Memo.LayoutMode = LayoutMode.wplayShowManualPageBreaks;

            // Activate the Syntax Higlighting for fields and bands also in
editor #2!
            wpdllInt2.Memo.TextCommandStr(16, 4, "");


            // OPTIONAL: Add some initial text
            wpdllInt1.TextCursor.InputString("Example\r", 0);
            wpdllInt1.TextCursor.InputString("<<#group>>\r", 0);
            wpdllInt1.TextCursor.InputString("<<:data \"databand\"/>>\r", 0);
            wpdllInt1.TextCursor.InputString("<<Field/>>\r", 0);
            wpdllInt1.TextCursor.InputString("<<#/group>>\r", 0);

        }
```

We use the OnSelectedIndexChange event of the tabcontrol to initialize the reporting engine, convert the tokens into the internal template format and, if this worked without a problem, start the reporting:

```
        private void tabControl1_SelectedIndexChanged(object sender, EventArgs
e)
        {
            if (tabControl1.SelectedIndex == 1)
            {
                wpdllInt2.Memo2.Clear(false, false);

                // Copy the text from Editor #1 to Editor #2 (on second page)
                object currtext = wpdllInt1.Memo.SaveToVar(false, "WPT");
                wpdllInt2.Memo.LoadFromVar(currtext, false, "AUTO");

                // Now do the auto conversion
                string res = wpdllInt2.Memo.TextCommandStr(17, 0, "");
                if (res != null)
                {
                    // Show an error message
                    MessageBox.Show(res);
                }
                else
                {
                    // Start reporting - fills any field with random data
                    // and loops any group 10 times
                    wpdllInt2.Report.Command(2); // Copy Header+Footer + Create
Report
                }

            }
        }
```

Please note that we use Report.Command(2) instead of Report.CreateReport to start reporting. This command will also copy any header and footer texts to the destination which can be desirable if the source template was a standard document with tokens. But without controlling events not much will happen, only the text will be copied from the template to the destination.

First we need an event to control the groups:

```
        // This event is used to start, prepare and close a report group
        private void wpdllInt2_OnReportState(object Sender,
            string Name, int State, IWPReportBand Band, ref bool Abort)
        {
            if(State==0) //REP_BeforeProcessGroup
                {
                    // Take Name to open a data base query. Move to first
Record
                    // and set Abort to TRUE if the query is empty!
                    // Here also Set total variables to 0.
                    Abort = Band.Count>10;
                }
        }
```

Please see "OnReportState" for a complete C# template.

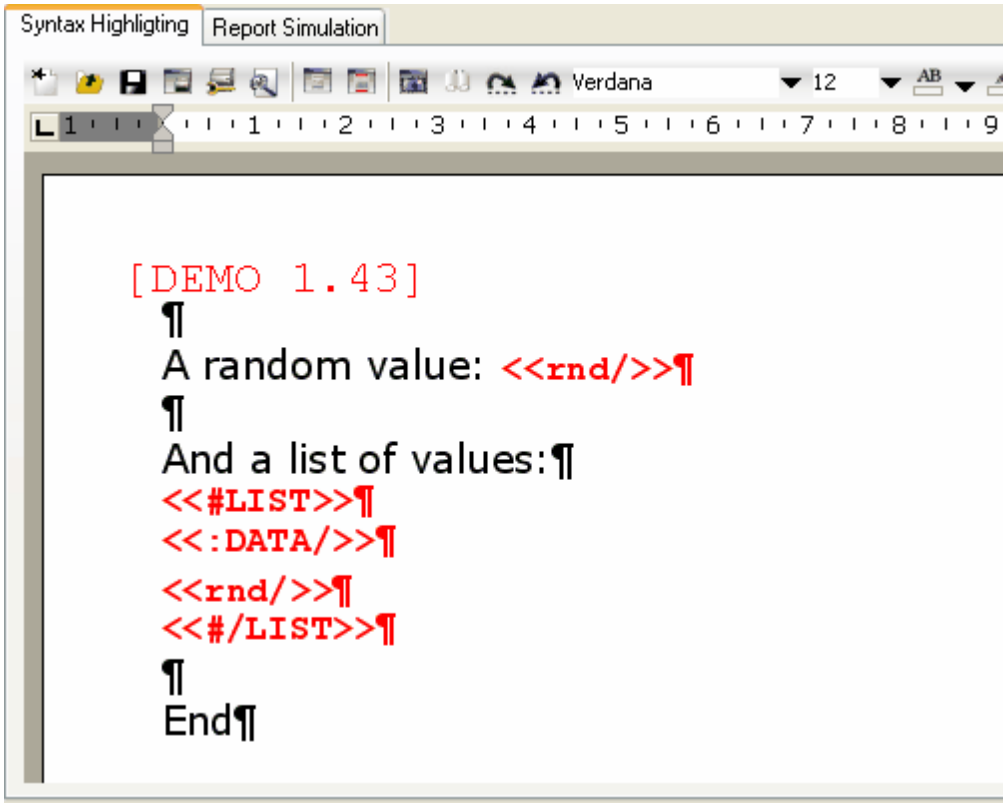
Now some code to fill in data for our fields is missing. To do so the event OnFieldGetText is used. Here we would usually read out a dataset, but for simplicity we here just insert a random value:

```
         Random rnd = new Random();
```

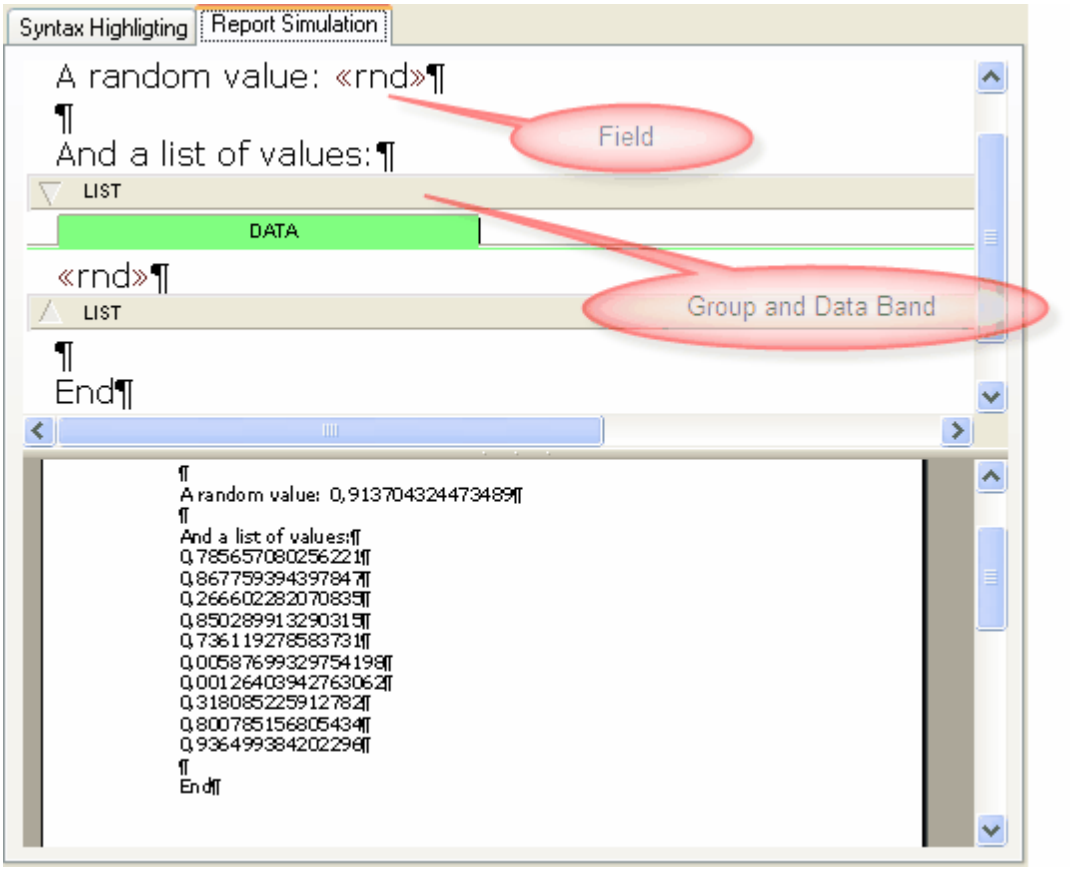
```
private void wpdllInt2_OnFieldGetText(object Sender,
          int Editor, string FieldName, IWPFieldContents Contents)
{
    Contents.StringValue = Convert.ToString(rnd.NextDouble());
}
```

Now we can already test reporting:

a) Template (with activated syntax highlighting)

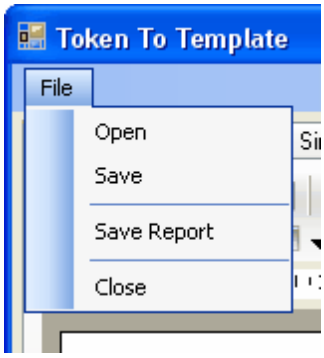


b) Second page
top editor = template in internal format
bottom editor = result



4.9.4.2 File Menu

Add a file menu:



We use this OnClick event handler:

```
private void Open_MenuItem_Click(object sender, EventArgs e)
  {
      wpdllInt1.wpaProcess("DiaOpen", "");
  }

private void Save_MenuItem_Click(object sender, EventArgs e)
  {
      wpdllInt1.wpaProcess("DiaSaveAs", "");
  }

private void Save_Report_MenuItem_Click(object sender, EventArgs e)
  {
```

```
            wpdllInt2.Memo2.wpaProcess("DiaSaveAs", "");
        }

        private void Close_MenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }
```

4.9.4.3 Use conditions for bands

TextDynamic allows simple conditions for bands. This makes it possible to use some parts of the text depending on the value of a certain field. Of course the field name can be virtual, not existing in the database but known by your code.

To use a condition in a band use the option:

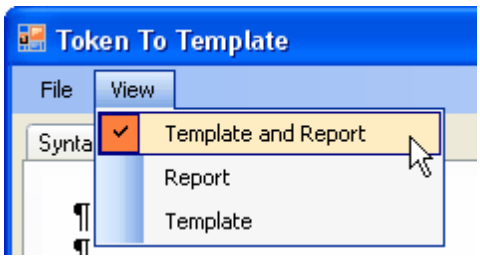
?fieldname=yyy (see Parameters for Bands and Groups)

To evaluate the specified fieldname the event OnFieldGetText is called. There the property Contents.IsMergeField will be false. Use ExecStrCommand the special states (null, zero, defined) of a field can be specified.

```
        private void wpdllInt2_OnFieldGetText(object Sender,
                  int Editor, string FieldName, IWPFieldContents Contents)
        {
            // test the conditions
            if (!Contents.IsMergefield)
            {
                if (FieldName == "ISNULL")
                    Contents.ExecStrCommand(1, 2, "");
                else if (FieldName == "ISZERO")
                    Contents.ExecStrCommand(1, 3, "");
                else if (FieldName == "ISUNDEF")
                    Contents.ExecStrCommand(1, 1, "");
                else
                {
                    Contents.ExecStrCommand(1, 4, ""); // Known
                    Contents.StringValue = "123";
                }
            }
            else Contents.StringValue = Convert.ToString(rnd.NextDouble());
        }
```

4.9.4.4 View Menu - (Hide/ShowTemplate)

We want to create a menu to show and hide the template and the created report.



This is the code for the 3 menu items. It uses the property Memo.Hidden.

```
        private void templateAndReportToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            wpdllInt2.Memo2.Hidden = false;
            wpdllInt2.Memo.Hidden = false;
            templateAndReportToolStripMenuItem.Checked = true;
            reportToolStripMenuItem.Checked = false;
            templateToolStripMenuItem.Checked = false;
        }

        private void reportToolStripMenuItem_Click(object sender, EventArgs e)
        {
            wpdllInt2.Memo2.Hidden = false;
            wpdllInt2.Memo.Hidden = true;
            templateAndReportToolStripMenuItem.Checked = false;
            reportToolStripMenuItem.Checked = true;
            templateToolStripMenuItem.Checked = false;
        }

        private void templateToolStripMenuItem_Click(object sender, EventArgs e)
        {
            wpdllInt2.Memo.Hidden = false;
            wpdllInt2.Memo2.Hidden = true;
            templateAndReportToolStripMenuItem.Checked = false;
            reportToolStripMenuItem.Checked = false;
            templateToolStripMenuItem.Checked = true;
        }
```

4.9.5 RTF2PDF - Reporting C# Sample

The principle is very similar when using our product the RTF2PDF TextDynamic Server.

Here this feature is used best with the token conversion or complete and tested templates. You can use TextDynamic to create and edit such templates.

```
using wPDF;
using WPDynamic;

private void ConvertTest_Click(object sender, System.EventArgs e)
{
   // Set License with Reportion Option
   rtF2PDF2.SetLicense( "---", "---", 0);

   // Load a File
   rtF2PDF2.Memo.LoadFromFile(@"c:\Template_With_Tokens.RTF", false, "");

   // Use the Token To Template Conversion
   rtF2PDF2.Memo.TextCommandStr(17,0, "");

   // Create the Report (will be in Memo2)
   rtF2PDF2.Report.CreateReport();

   // Set Output Filename
   rtF2PDF2.FileName =  @"c:\test.pdf";

   // And convert Memo2 to PDF
```

```
   rtF2PDF2.PrintSecond();
}
```

We are also using this event to loop each group 10 times:

```
private void rtF2PDF2_OnReportState(object Sender, string Name, int State, WPDynamic.IWPR
{
      if (State==0)
      Abort = Band.Count>10;
}
```

4.9.6 DB-Description and Template architecture

When speaking of "reporting" we mean the process that data which comes from a database or has been calculated is merged with the text and layout objects stored in a report template to create a new document. Already the basis version of TextDynamic has a powerful method to mix text and data, called mail merge. In contrast to this easy to use method the "reporting" we talk about here allows that areas of the report template can be looped for each row in the input data which is typically the result of a database query.

Using the "Token to Template Conversion" it is possible to create powerful templates which can be used to create complicated documents. But since it is based on simple text it does not offer, apart from syntax highlighting, much possibilities to detect errors in a template (for example a data field defined by a sub query is used outside of the group which enumerates the records in this query).

To be able to detect such errors it is required for the editor to know the underlying database structure. Since TextDynamic does not implement database access (except for DAO) the structure must be made known using a special description, the "DB-Description".

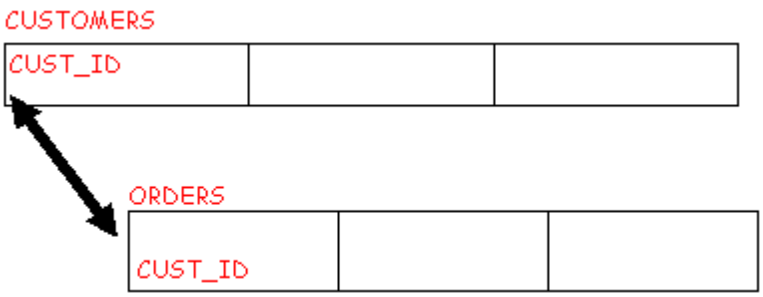
To make it possible

Of course the looping allows recursion to work with relationally organized data.

Example of a customer organization application:

We need a database with two tables, one table contains the names and addresses of the customers of a company. Each customer has a unique number: "CUST_ID".

The second table contains all orders received by the sales department. To identify that a certain order was placed by a certain customer the (lookup-)field CUST_ID receives the value of the field CUST_ID found in the first table, the customer table.



In a real application we would need additional tables to group orders by dates, to store received payments, to cancel orders and so on. But we want to keep things simple, only show the basic concepts.

Creating the data input form for the two tables would be fairly simple, but the code which creates and prints the invoice can be tricky.

TextDynamic simplifies this problem for you by separating the necessary tasks into several steps.

(a) DB-Description (XML)

Collect the required fields from the involved tables and add into virtual groups.

What is done here is basically the creation of a XML structure which represents the graphic above. In this step you also add group variables which are used to sum up values to create totals.

The XML DB-Description can be edited as text and loaded into TextDynamic or you can use a special API to build the template. We added a feature to auto create such a template from DAO compatible databases, such as MS Access. (See Example)

The DB-Description is maintained by the IWPReporter interface.

(b) Template (RTF with extensions)

From the XML structure created in (a) TextDynamic can create and maintain a raw reporting template.
The devloper can now format this template, add or remove fields and text.
You can also enable the end users to adjust the reporting template to their needs.
An external application is not required for this, all is done with TextDynamic.

(c) Events used to retrieve and locate data

Create the necessary programming logic to provide the data for the groups. Usually you will create SQL queries for the inner groups each time one row of the outer group is processed.
(Although SQL allows the GROUPBY mode in the SELECT method, we recommend to execute multiple SQL statements for the inner groups. So you have more control over the ordering of the rows and deeper nesting levels are no problem)

4.9.7 MS Access Example

In this topic we

1) show how to use the events to control the creating of a report in MS Access

2) Show how to use the DAO Interface to read and convert a MS Access Database automatically

4.9.7.1 MS Access code

In this chapter we show some real word MS Access code to handle the reporting events.

Event to control how often a group is processed

```
' This event is used to control how often a group will be processed
' The most important values of parameter 'State' are:
' 0=BeforeProcessGroup - check if we are at EOF
' 8=AfterProcessGroup - move to next record
Private Sub WPDLLInt1_OnReportState(ByVal NAME As String, ByVal State As Long, ByVal Band
As Object, Abort As Boolean)
  If State = 0 Then
     Abort = RepRS.EOF
```

```
  End If
  If State = 8 Then
     RepRS.MoveNext
     Abort = RepRS.EOF
  End If
End Sub
```

Event to read the text of a field

```
' This event is used to retrieve the value of a certain field
Private Sub WPDLLInt1_OnFieldGetText(ByVal Editor As Long, ByVal FieldName As String, ByVal
Contents As Object)
  Dim ContentsObj As IWPFieldContents
  Dim dn As String
  Dim fn As String
  Dim i As Integer
  Dim j As Integer
  Set ContentsObj = Contents
  j = -1
  dn = StripDBName(FieldName, fn)
  For i = 0 To RepRS.Fields.Count - 1
    If (RepRS.Fields(i).SourceTable = dn) And _
       (RepRS.Fields(i).SourceField = fn) Then
         j = i
    End If
  Next

  If j >= 0 Then
     ContentsObj.StringValue = RepRS.Fields(j).Value
  Else
    ' DO NOT ASSIGN ANYTHING HERE
    ' Otherwise Variables are not found
    ' ContentsObj.StringValue = 'unknown:' + FieldName
  End If
End Sub
```

Event to read the value of a variable (used in formulas)

```
Private Sub WPDLLInt1_OnReadFormulaVar(ByVal FieldName As String, ByVal GroupContext As
Object, ByVal BandContext As Long, Value As Double)
  Dim dn As String
  Dim fn As String
  Dim i As Integer
  Dim j As Integer
  j = -1
  dn = StripDBName(FieldName, fn)
  For i = 0 To RepRS.Fields.Count - 1
    If (RepRS.Fields(i).SourceTable = dn) And _
       (RepRS.Fields(i).SourceField = fn) Then
         j = i
    End If
  Next
  If j >= 0 Then
     Value = RepRS.Fields(j).Value
  End If
End Sub
```

The last 2 event handler require this utility function

```
Private Function StripDBName(aName As String, ByRef aFieldName As String) As String
  Dim i As Integer
  Dim DBName As String
  Dim f As String
  DBName = ""
  i = InStr(aName, ".")
```

```
  If i > 0 Then
    DBName = Left$(aName, i - 1)
    aFieldName = Mid$(aName, i + 1)
  End If
  StripDBName = DBName
End Function
```

Also required are 2 global variables

```
Dim RepRS As Recordset
Dim RepRSName As String
```

The report is started using this code

```
Private Sub Create_Report_Click()
  Const strQueryName = "ORDERS Query"
  Dim db As Database
  Dim td As WPDLLInt
  Set td = WPDLLInt1.Object
  Set db = CurrentDb() ' Open pointer to current database
  RepRSName = "ORDERS"
  Set RepRS = db.OpenRecordset(strQueryName) ' Open recordset on saved query

  ' Now Create the report. MoveNext and EOF is used in events!
  td.Report.CreateReport

  RepRS.Close
  db.Close
End Sub
```

4.9.7.2 DAO Example to create DB-Description

To start simple we only create a list from items in table "Orders". This would be already sufficient to create an invoice for a certain customer.

We create nested groups for each of the tables. Inside each group we add the possible fields.

The field names can be usually read easily from the table object. After the structure was created it is used to create a reporting template. The user can modify the template and add fields which are listed in the "repository".

Example written in MS Access:

Initialization of the editor:

```
Private Sub Form_Load()
  ' Initialize License
  ' WPDLLInt1.EditorStart 'licensename', 'licensekey'

  ' load the PCC file
  WPDLLInt1.SetLayout ".\buttons.pcc", "default", "", "main", "main"
  ' We need the 'double editor' mode, toolbar, tables and reporting
  ' If this initialization is missing 'Set Report = td.Report' will fail
  ' Editor 1 gets a ruler and scrollbars
  ' Editor 2 gets scrollbars, navigation and view panels
  WPDLLInt1.SetEditorMode 1, 2 + 8 + 16 + 32, 2 + 4 + 8, 4 + 8 + 128 + 258
  ' Select normal page layout
  WPDLLInt1.SetLayoutMode 0, 0, 100
End Sub
```

Procedure which takes one database table and creates a simple template to list all fields except for "ID"

```
Private Sub Recreate_Template_Click()
Dim td As WPDLLInt
Dim Report As IWPReport
Dim db As DAO.Database
Dim tdf As DAO.Recordset
Dim i As Integer
Dim mode As Integer
Dim tblNameToLoop As String
Dim fieldName As String
Dim fieldCount As Integer
Set db = CurrentDb() 'pointer to current database
Set td = WPDLLInt1.Object

Set Report = td.Report
' the next line fails if reporting as not been activated in Form_Load()
Report.Clear

' Create a table of all fields in table 'CUSTOMERS'
tblNameToLoop = "ORDERS"
Report.AddGroup tblNameToLoop, "", tblNameToLoop, "*", 0, ""
Set tdf = db.OpenRecordset(tblNameToLoop, dbOpenSnapshot, dbSeeChanges)
fieldCount = tdf.Fields.Count
For i = 0 To fieldCount - 1
  fieldName = tdf.Fields(i).NAME
  ' we can deselect some of the fields which are used for the table relation
  If InStr(fieldName, "ID") > 0 Then
     mode = 2 ' Deselected but visible.
     Else
     mode = 0 ' Standard: Visible and Selected
  End If
    Report.AddField tblNameToLoop + "." + fieldName, fieldName, "", _
    "*", "", "", "", 0, mode, 0
Next
tdf.Close

' Add group variables here (if required)
' ...

' We need bands, otherwise the group is empty!
Report.AddBand "*", "Header", 1, 0, "", "", 0, 0
Report.AddBand "*", "Data", 0, 0, "", "", 0, 0
Report.AddBand "*", "Footer", 2, 0, "", "", 0, 0

db.Close

' Init the repository and create a reporting template
Report.InitTemplate "@CUSTOMERS LIST", 5
End Sub
```

If you need calculation, for example to create an invoice you can add group variables.

```
' Group Variable to calculate the price in each row
Report.AddVar "ORDER_PRICE", "Sub Total", "Price * Amount", "*", _
"=0", "CUR=$", "", 0, "=ORDERS.AMOUNT*ORDERS.PRICE", 0
' Group Variable to calculate the total price, note the '+=' and mode=2
' This variable should be use in the footer
Report.AddVar "TOTAL_PRICE", "Total", "Sum of all ORDER_PRICE", "*", _
"=0", "CUR=$", "", 2, "+=ORDERS.AMOUNT*ORDERS.PRICE", 0
```

Group variables are processed before each time the group is used. The StartFormula is only

executed before the first time. By default a variable has the value 0 (the formula "=0" is redundant). In the LoopFormula use += to sum up values.
References to table values are possible in formulas, for example "ORDERS.AMOUNT". Since TextDynamic does not access the database directly, the value must be provided using the event OnReadFormulaVar.
Note: If in a report template a field uses the name of a variable, that variable is assigned.

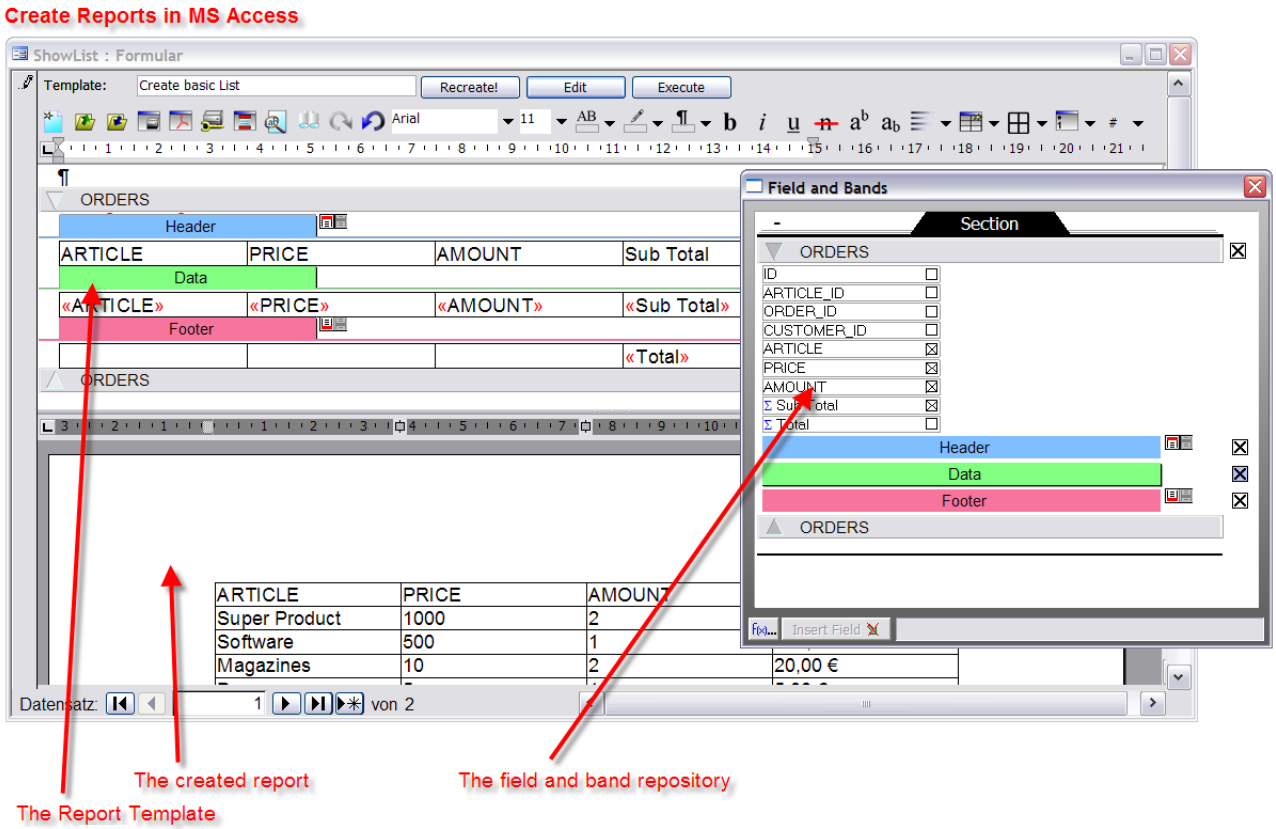
To show the report editor without recreation of the template use this code:

```
Private Sub Edit_Template_Click()
  Dim td As WPDLLInt
  Set td = WPDLLInt1.Object
  td.Report.InitTemplate "@Field and Bands", 6
End Sub
```

Create the Report in MS Access

Using the **integrated support for DAO interfaces**, a list can be created with just a few additional lines. Basically only a record set has to be created and assigned to Report.Recordset.

```
Private Sub Create_Report_Click()
  Const strQueryName = "ORDERS Query"
  Dim db As Database
  Dim td As WPDLLInt
  Dim RepRS As Recordset
  Set td = WPDLLInt1.Object
  Set db = CurrentDb() ' Open pointer to current database
  Set RepRS = db.OpenRecordset(strQueryName) ' Open recordset on saved query

  td.Report.Recordset = RepRS
  If td.Report.SetAutomatic(1, "") Then
     td.Report.CreateReport
  End If

  RepRS.Close
  db.Close
End Sub
```



4.9.8 API

Interfaces	Methods
IWPReport	WPDLLInt.CreateReport
IWPReportBand	WPDLLInt.Report
IWPReportVar	IWPTextCursor.ReportConvertTable
Events	IWPTextCursor.ReportConvertText
OnFieldGetText	IWPTextCursor.ReportInputBand
OnReadFormulaVar	IWPTextCursor.ReportInputGroup
OnReportState	

4.9.8.1 IWPReport

Description

This interface can be accessed through the property Report. It allows the preparation of a reporting template definition with obligatory and optional groups, bands and fields.

```
Please don't forget to activate the double editor and the reporting support:
  SetEditorMode(
        EditorMode.wpmodDoubleEditor,
        EditorXMode.wpmodexReporting|...)
The second editor can be hidden using: wpdllInt1.Memo2.Hidden = true;
```

Manage report template and reporting.

Please note the new token to template conversion. It allows it to convert simple text into a reporting template and is suitable to solve most of the reporting problems.

Here we discuss the more advanced techniques. Please first read the chapter "Reporting".

In this case you will only need this methods:

IWPReport.CreateReport
IWPReport.Command

Please also see the description of this events:
OnFieldGetText
OnReportState

Overview

a) This interface has basic functions to initialize the template editor and start reporting.
b) The property RecordSet makes it easy to create a list in MS Access by using a DAO RecordSet interface.
c) This interface manages an internal XML DB-Description. This DB-Description is used to create a reporting template from all groups, bands and fields which are marked to be obligatory. Optional groups and bands can be selected by the user. A dialog box will show the available elements in this DB-Description as drag&drop repository. Internally the template is managed as XML file - it can also be saved and loaded in this format.

Please see the introduction "Reporting".

The DB-Description is internally managed as a XML structure, it can be loaded from XML and saved to XML. When the XML structure is used to create a reporting template, a copy of the XML data is attached to the template.

Report.InitTemplate(Filename,Mode)

By simply calling the method InitTemplate the template is prepared in the upper editor. The user can then add optional fields, adjust text attributes or type in some text.

InitTemplate function parameters:

Filename: This is an optional filename. If specified the template (NOT the xml data!) will be loaded from this file. This can be usefull to load the initial paragraph style list. If starting with "@", this parameter is used to set the caption for the DB-Description form.

Mode: If **bit 1** is set, the template in the editor #1 will be initialized using the XML data (which was loaded or created). Otherwise, in case a file name was specified, only the report template will be loaded. In this case it is recommended to also set bit 2 to avoid out of sync XML data.
If **bit 2** is set, the engine will load the XML from the report template and overwrite the information stored in the report object.
If **bit 3** is set, the DB-Description form (also called repository) will be displayed. Once this form was displayed the methods ShowTemplate and ShowResult will automatically show and hide this form.

Examples:

Load a DB-Description and create a raw report template
Report.LoadFromFile("c:\\groups_**and**_fields.xml");

```
Report.InitTemplate("@A new report",1+4);
```

Load a previously created template (RTF or WPT format) & show the repository form

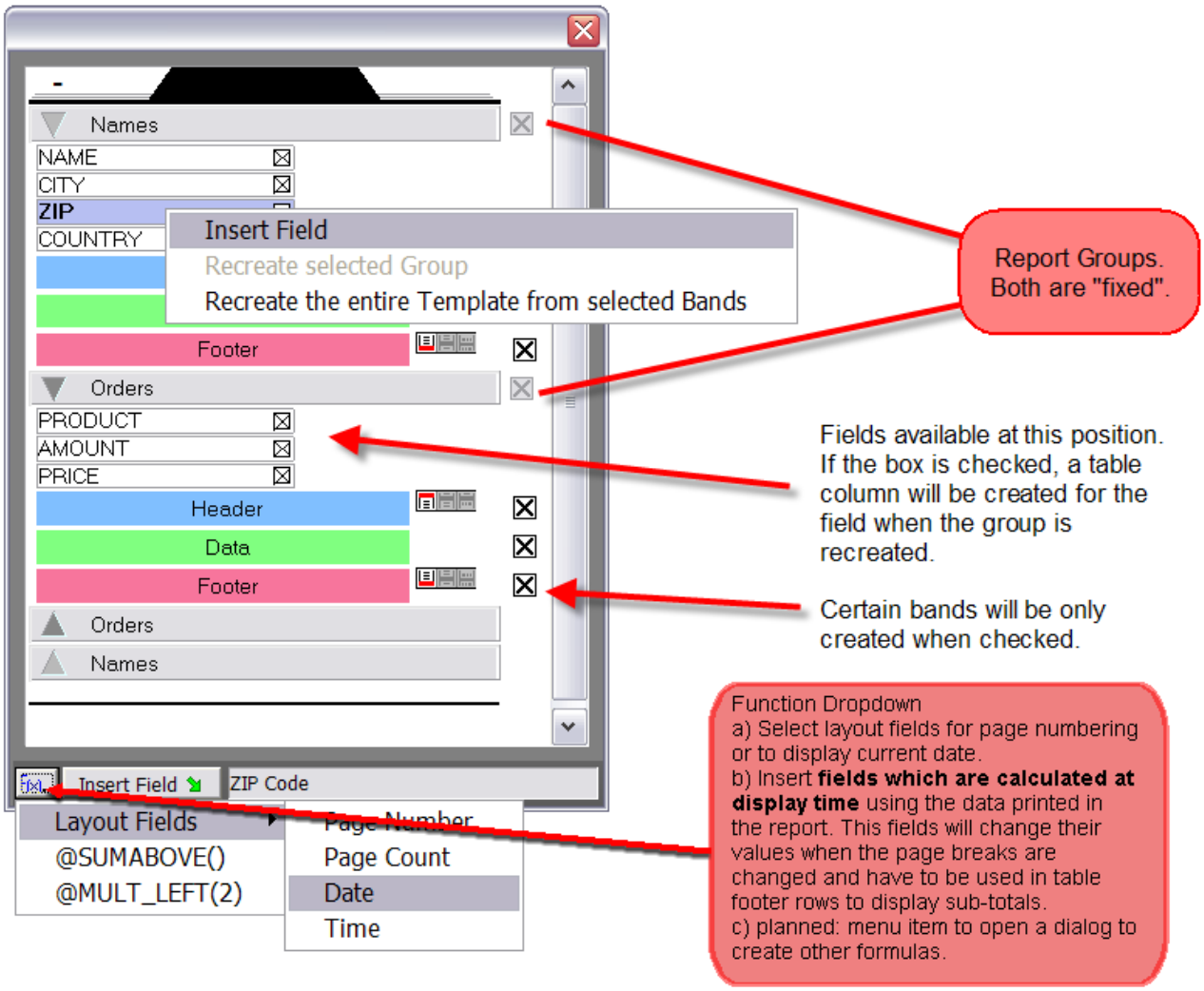
```
Report.InitTemplate("c:\\a_report_template.rtf",2+4);
```

Only set caption for DB-Description form (must be already displayed)

```
Report.InitTemplate("@a new caption",0);
```

"Repository Form":

This special dialog can be displayed in stay-on-top-mode to insert fields and to update the template using a changed selection of the optional bands. This dialog internally updates the XML structure, so the selection of bands and fields can also be saved back to XML.



Properties	Methods
AsXML	AddBand
BandCount	AddField
Database	AddGroup

RecordSet
RepositoryCaption
ShowRepository
TemplateEditorStyle

AddSection
AddText
AddVar
Band
Clear
Command
CreateReport
DefinePageSize
FindGroup
GetError
GetErrorCount
InitGroup
InitTemplate
LoadFromFile
ModifyXML
SaveToFile
SelectSection
SetAutomatic
SetProp
ShowResult
ShowTemplate

4.9.8.1 A) XML DB-Description

The interface "Reporter" manages a DB-Description defined in XML. The XML structure is used to define and validate the variables.

Please don't mix up this DB-Description with the template used by the "Token to Template Conversion". The first is a true XML data while the latter is formatted text with added tokens to mark fields and bands.

This is the XML data which was created in the first example. The XML data can be simply saved using Report.SaveToFile, edited in a separate application and loaded back in using LoadFromFile.

```
<?xml version="1.0" encoding="windows-1252"?>
<main Type="Section">
 <group Name="ORDERS" Title="ORDERS" Mode="0">
  <field Name="ORDERS.ID" Title="ID" Mode="2"/>
  <field Name="ORDERS.ARTICLE_ID" Title="ARTICLE_ID" Mode="2"/>
  <field Name="ORDERS.ORDER_ID" Title="ORDER_ID" Mode="2"/>
  <field Name="ORDERS.CUSTOMER_ID" Title="CUSTOMER_ID" Mode="2"/>
  <field Name="ORDERS.ARTICLE" Title="ARTICLE"/>
  <field Name="ORDERS.PRICE" Title="PRICE"/>
  <field Name="ORDERS.AMOUNT" Title="AMOUNT"/>
  <var Name="ORDER_PRICE"
       Title="Sub Total"
       Description="Price * Amount"
       StartFormula="=0"
       LoopFormula="=ORDERS.AMOUNT*ORDERS.PRICE"
```

```
        Format="CUR=$"/>
  <var Name="TOTAL_PRICE"
        Title="Total"
        Description="Sum of all ORDER_PRICE"
        StartFormula="=0"
        LoopFormula="+=ORDERS.AMOUNT*ORDERS.PRICE"
        Format="CUR=$"
        Mode="2"/>
  <header Title="Header"/>
  <data Title="Data"/>
  <footer Title="Footer"/>
  </group>
</main>
```

In the XML structure the possible group nesting is outlined. Also fields which are available are defined. Optionally calculated fields ("var") to create totals can be also added. The XML structure also contains information for the creation of a default reporting template. Here the obligatory groups and bands are created (Mode). Text can be filled in using HTML code. The captions use the 'Titel' defined for a field. (So changing just one element will update all references)

The XML structure can be created using program code. At best by reading the structure of a database connection and reusing the "Titel" and information of the field. To do so the Reporter interface has several procedures to add groups, fields and bands. It is also possible to add bands and fields to an existing template.

Once the default template has been created it can still be edited in the regular text editor.

4.9.8.1 B) What are groups

Groups are used in reports to mark areas of the text which is used as long as there is data to export. The event OnReportState is used to check for the end condition.

4.9.8.1 C) What are bands

Bands can be data bands, header or footer bands.

Headers are used to mark text parts (usually a table row) which should be written before the group data and, optionally, at the top of each page.

Footers are used to mark text parts (usually a table row) which should be written after the group data and, optionally, at the bottom of of each page. The latter is a feature unique to TextDynamic.

Databands mark the part of the group which is used for each data row.

4.9.8.1 D) What are fields

Fields are placeholders which are filled with using the event OnFieldGetText.Usually a field receives the contents of exactly one data base field, but the code behind OnFieldGetText can also clalculate a number, concatenate strings or even insert images or tables inside the field.

Fields can also be defined by a formula. Here the OnFieldGetText event is not beeing triggered, instead the formula is evaluated. (see more below)

4.9.8.1 E) What are group variables

Group variables are used to calculate values while the report is beeing processed. The variables can also be inserted into the report template, usually the footer row, to retrieve the stored value.

Variables are always inserted as text-objects, not as merge variables in the text.

Variables marked as hidden (mode=16) cannot be inserted in the report. They can only be referenced by fields which read out the value or use it to create a chart.

4.9.8.1 F) What are formulas

Formulas are used to calculate totals. TextDynamic only allows numeric formulas. If you need to combine string fields you need to create a new field name which represents the combined value of two physical fields.

Inside formulas variables are allowed, i.e. "VARA+VARB". Here the event OnReadFormulaVar is used to retrieve the value which should be inserted instead of VARA and VARB.

Some special formulas are allowed at certain places:

If a group variable uses the LoopFormula "add(field(fieldname),formula)" first the event OnFieldGetText will be triggered for "fieldname" and then the formula will be evaluated. The result and the text retrieved for the field will be added to the element list of the variable. This list can used later to create a chart. The formula can be also written as add(fieldname,formula). "formula" must be a string constant.

Fields can also read out the value of a group variable of a previous, higher level group. To do so specify the name of the group variable in the formula using "var(varname)".

4.9.8.1 G) Properties

? AsXML

Applies to
IWPReport

Declaration
```
string AsXML;
```

Description
The DB-Description data as XML string.

? BandCount

Applies to
IWPReport

Declaration
```
int BandCount;
```

Description
Count of bands in the highest level in the report template (this is in editor 1, it is not the XML data!). To get access to a band or group use GetBand.

This property is readonly.

? Database

Applies to
IWPReport

Declaration
```
IUnknown Database;
```

Description
This property is not used. You can use it to store your connection.

? RecordSet

Applies to
IWPReport

Declaration
```
IUnknown RecordSet;
```

Description
You can assign a DAO.RecordSet interface to this property. Then use Report.SetAutomatic(1, "") to activate the automatic mode.

MS-Access Example:

```
Private Sub Create_Report_Click()
  Const strQueryName = "ORDERS Query"
  Dim db As Database
  Dim td As WPDLLInt
  Dim RepRS As Recordset
  Set td = WPDLLInt1.Object
  Set db = CurrentDb() ' Open pointer to current database
  Set RepRS = db.OpenRecordset(strQueryName) ' Open recordset on saved query
  td.Report.Recordset = RepRS
  If td.Report.SetAutomatic(1, "") Then
     td.Report.CreateReport
  End If
  RepRS.Close
  db.Close
End Sub
```

? RepositoryCaption

Applies to
IWPReport

Declaration
```
string RepositoryCaption;
```

Description
Caption of the repository form.

? ShowRepository

Applies to
IWPReport

Declaration
```
bool ShowRepository;
```

Description
Display the repository.

? TemplateEditorStyle

Applies to
IWPReport

Declaration
```
int TemplateEditorStyle;
```

Description
Currently not used - must be 0.

4.9.8.1 H) Methods

? IWPReport.ConvertTokens

Converts the document in the editor into a reporting template

See: Token To Template Conversion

? IWPReport.CreateReport

Applies to
IWPReport

Declaration
```
int CreateReport();
```

Description
This method Starts the report process.

The second editor is cleared and the page size is copied from the the first editor which holds the template.

More features are available through Command(1 - 3).

? IWPReport.Command

Applies to
IWPReport

Declaration
```
int Command(int ID);
```

Description

Command ID =1:

Start the report, works like CreateReport.

Command ID =2:

Start the report like CreateReport but first copies the header and footer for first/odd and even pages to the destination. The body is cleared.

Command ID =3:

Start the report like CreateReport but does not clear the destination. The text is simply appended.

? IWPReport.AddBand

Create a band definition in the DB-Description.

Applies to
IWPReport

Declaration

```
procedure AddBand(const Group: WideString; const Title: WideString; BandType:
Integer; ColumnCount: Integer; const FieldList: WideString; const ParStyle:
WideString; RadioGroup: Integer; Mode: Integer);
```

Description
You can use AddBand to build a DB-Description (which can be saved and loaded in XML format). The use of DB-Descriptions makes it easier for the end user to manage the report template since the technical details are hidden.

Parameters	
Group	The name of the parent group. Use "*" to select current group. An empty string will insert band at the top level.
Title	The Title of the band. This will be displayed in the template editor.
BandType	Selects the band type and the visibility: bit 1(=1): Creates a header band. bit 2(=2): Creates a footer band. The bits 3-5 are only used for headers and footers in groups: bit 3(=4): Hide at start (headers) and end (footers) of group bit 4(=8): Also display after (headers) and before page breakes (footers). This bit is used to display sub totals in footers! We recommend to create two footers, one with bit 3 and 4 set, the other with both bits clear. bit 5(=16): reserved. The bits 6-7 are only used for header and footers at top level which are used to create document header and footers. If none of these bits is set, the band is always used: bit 6(=32): Hide text on first page.

	bit 7(=64): reserved
	bit 8(=128): Show text on odd pages only.
	bit 9(=256): Sow text on even pages only.
	bit 10(=512): Only use when selected (see RadioGroup)
	Example BandType Values:
	`BANDTYP_HEADER_ALL = 1;`
	`BANDTYP_FOOTER_ALL = 2;`
	`BANDTYP_HEADER_FIRST = BANDTYP_HEADER_`
	`BANDTYP_FOOTER_FIRST = BANDTYP_FOOTER_`
	`BANDTYP_FOOTER_LAST = BANDTYP_FOOTER_`
	`BANDTYP_OVERFLOW_HEADER = BANDTYP_HEADI`
	`BANDTYP_OVERFLOW_FOOTER = BANDTYP_FOOTI`
	Note
	When the DB-Description is saved to XML the bits 3 and 4 are converted into bit 1 and 2 of parameter "Usage". The bits 5 to 8 are converted into bit 1 to 4 of the parameter "Visibility"

ColumnCount	This is the count of columns which should be created. The value can be 0 if the fields of the group are used - see Mode. If the ColumnCount<0 no columns will be created and the text specified in string "FieldList" will be used to create regular text. "FieldList" can be also specified as HTML text using bit 4 in Mode.
FieldList	This is a comma separated list of the fields which should be used. Can be "" if the fields in the group are used.
ParStyle	Either "" or paragraph style for the table cells which are created.

RadioGroup	**"RadioGroup" is under development - reserved for future**
	If this value is not 0, a "radio band" will be created. Radio bands can be enabled or disabled in the template editor (not the repository!) and only the enabled band will be used during the reporting process. The group number is taken from the low word of this parameter. Enabling a band automatically disables all other bands which use the same number and are not "siblings". Siblings are bands which use the same group number in low word and the same sibling number > 0 in the high word. The BandType bit 10 is assumed when the value of RadioGroup is not 0.
	`Example:`
	`GROUP1 = 1;`
	`GROUP1_SIBLINGSA = 1 + $10000;`

Mode	This parameter controls if the band is selected and how the band is used to create a reporting

	template.
	bit 1: In the repository the user cannot change the selected state.
	bit 2: The band is **not** selected.
	bit 3: Do not create default columns when the the ColumnCount value is 0. Unless a field list has been specified, the "default columns" are created for all selected fields and variables in the same group.
	bit 4: Use the field text as HTML source. If this bit is set you can specify text and fields for each column.
	`Example:`
	`"<field name="Product"/>,<field name="Ar`
	bit 5: The band is hidden in the repository. It can still be selected to create a band in the report template.

? IWPReport.AddField

Applies to
IWPReport

Declaration
```
procedure AddField(const Name: WideString; const Title: WideString; const Description: WideString; const Group: WideString; const Formula: WideString; const Format: WideString; const ParStyle: WideString; FieldType: Integer; Mode: Integer; WidthTW: Integer);
```

Description
This method is used to create a new field in the XML description.

Parameters	
Name	The name of the field.
Title	The title of the field.
Description	The description what this field contains.
Group	The group to add to, use "*" for current group.
Formula	Currently not used.
Format	Currently not used.
ParStyle	The paragraph style to specify the character attributes.
FieldType	Currently not used - pass 0.
Mode	The mode bits, to select and hide the field.
WidthTW	This value is reserved.

? IWPReport.AddGroup

Applies to
IWPReport

Declaration
```
procedure AddGroup(const Name: WideString; const Alias: WideString; const Title: WideString; const Group: WideString; Mode: Integer; const DBParams: WideString);
```

Description
This menthod is used to start a new group. Groups contain bands, fields, variables and can also contain other groups. During the report creation a group is processed over and over again - as long as data is there to export.

Parameters	
Name	This is the name of this group.
Alias	This is an alternative name which can be used to select a certain data query.
Title	The title will be displayed in the headline of the group.
Group	This parameter is the name of the parent group. Please use "*" to select the current group or "" to create the group at top level.
Mode	This parameter contains bits to change the behavoir of the group. Bits 1 to 3 control the selection as described under AddBand. The bit 7 (=32) forces a page break before the group. Bit 8 forces it after the group. If bit 9 (=128) is set, the engine start a new table and does not try to continue a table after this band. Please note that this changes the behavior of formulas.
DBParams	This is an optional string parameter which can be used to create a query.

? IWPReport.AddSection

Applies to
IWPReport

Declaration
```
procedure AddSection(const Name: WideString; const Title: WideString; const PrintText: WideString; const ParStyle: WideString; const PageSize: WideString; Mode: Integer);
```

Description
Use this method to start a new section. Currently only one section is allowed.

? IWPReport.AddText

Applies to
IWPReport

Declaration
```
procedure AddText(const PrintText: WideString; const ParStyle: WideString);
```

Description
Creates an entry in the DB-Description to create a regular paragraph. Usually you will not have to use AddText - it is better to use the AddGroup, AddBand, AddField to create the logical structure of a report (can be saved as XML XML), then use InitTemplate to create a reporting template from the XML data. Now you can edit the reporting template using the TextDynamic editor and save it for later use.

Parameters	
PrintText	The text to display

ParStyle	The paragraph style

? IWPReport.AddVar

Applies to
IWPReport

Declaration
```
procedure AddVar(const Name: WideString; const Title: WideString; const Description: WideString; const Group: WideString; const StartFormula: WideString; const Format: WideString; const ParStyle: WideString; Mode: Integer; const LoopFormula: WideString; const GetTextForumula: WideString; WidthTW: Integer);
```

Description
This method adds a variable which can be added to the report template, just like a field. In contrast to a field a variable can only represent a number value. It is designed to keep the value between different loops of the same report group - therefore it can be used to create totals. Its value is modified by the assigned formulas "StartFormula" and "LoopFormula". The first formula is evaluated when the group the field is owned by is used first, the second formula is before each time (including the first) the group is used. This means that when the text of a group is evaluated the variables are already processed. Both the "StartFormula" and "LoopFormula" should be written as assignment to modify the value. (Exception function "add ()").
Example: LoopFormula = "+=ORDERS.AMOUNT*ORDERS.PRICE".

Parameters	
Name	The name of the variable
Title	The title of the variable
Description	The description what this variable does
Group	The group to add to, use "*" for current.
StartFormula	This formula is used to assign the start value. If empty or no assignment (not =, += etc) the initial value is 0.
Format	A string to specify the print format, use "CUR=€" use print EUROS, or "CUR=$" for Dollars.
ParStyle	The paragraph style to specify the character attributes.
Mode	The mode bits, to select and hide the variable.
LoopFormula	This formula is evaluated before each loop of the group, including first. Use "+=" to sum up a value or "add(field('orders.name'), orders. price)" to add a name which can be used in a chart. (not yet implemented)
GetTextFormula	This formula is used to create the text for the field.
WidthTW	This value is reserved.

? IWPReport.Band

Applies to
IWPReport

Declaration

```
function Band(Index: Integer): IWPReportBand;
```

Description
Retrieve the interface of one of the bands at the highes level in the template editor (#1). This can be used to change the properties of bands using code. Also see Band.Band() and property BandCount.

This C# example recursively loops through all the bands in the report and prints the names in Editor #2.

```
private void listband(IWPReportBand band, int d)
{
 int i;
 int bc = band.BandCount;
 for (i = 0; i < d; i++)
  wpdllInt1.Memo2.TextCursor.InputString("  ",0);
 wpdllInt1.Memo2.TextCursor.InputString(
  band.Typ + "  " +  band.Name + "\r",0);
 for (i = 0; i < bc; i++)
   listband(band.Band(i),d+1);
}
private void ListBands_Click(object sender, System.EventArgs e)
{
 IWPReport Report = wpdllInt1.Report;
 int bc = Report.BandCount;
 wpdllInt1.Memo2.Clear(false,true);
 for (int i = 0; i < bc; i++)
    listband(Report.Band(i),0);
 wpdllInt1.Memo2.ReformatAll(false,true);
}
```

? IWPReport.Clear

Applies to
IWPReport

Declaration
```
procedure Clear;
```

Description
Clears the XML DB-Description.

? IWPReport.DefinePageSize

Applies to
IWPReport

Declaration
```
procedure DefinePageSize(const Name: WideString; Width: Integer; Height:
Integer; TopMargin: Integer; BottomMargin: Integer; LeftMargin: Integer;
RightMargin: Integer; Mode: Integer);
```

? IWPReport.FindGroup

Applies to
IWPReport

Declaration
```
function FindGroup(const Name: WideString): IWPReportBand;
```

Description
Finds a group anywhere in the template (editor one).

? IWPReport.GetError

Applies to
IWPReport

Declaration
```
function GetError(Index: Integer): WideString;
```

Description
Retrieve an error message.

? IWPReport.GetErrorCount

Applies to
IWPReport

Declaration
```
function GetErrorCount: Integer;
```

Description
How many errors happend during reporting.

? IWPReport.InitGroup

Applies to
IWPReport

Declaration
```
procedure InitGroup(const GroupName: WideString; const AfterGroup: WideString);
```

? IWPReport.InitTemplate

Applies to
IWPReport

Declaration
```
procedure InitTemplate(const filename: WideString; Mode: Integer);
```

Description
This is an optional filename. If specified the template (NOT the xml data!) will be loaded from this file. This can be usefull to load the initial paragraph style list. If starting with "@", this parameter is used to set the caption for the DB-Description form.

Parameters	
Filename	Can be a file name or the caption for the "repository".
Mode	If bit 1 is set in parameter mode, the template in the editor #1 will be initialized using the XML

	data (which was loaded or created). Otherwise, in case a file name was specified, only the report template will be loaded. In this case it is recommended to also set bit 2 to avoid out of sync XML data. If bit 2 is set, the engine will load the XML from the report template and overwrite the information stored in the report object. If bit 3 is set, the DB-Description form (also called repository) will be displayed. Once this form was displayed the methods ShowTemplate and ShowResult will automatically show and hide this form.

? IWPReport.LoadFromFile

Applies to
IWPReport

Declaration
```
function LoadFromFile(const filename: WideString): WordBool;
```

? IWPReport.ModifyXML

Applies to
IWPReport

Declaration
```
procedure ModifyXML(const Path: WideString; const Name: WideString; const param:
WideString; const ParamValue: WideString);
```

Description
Reserved

? IWPReport.SaveToFile

Applies to
IWPReport

Declaration
```
procedure SaveToFile(const filename: WideString);
```

Description
Saves the XML data to a file.

? IWPReport.SelectSection

Applies to
IWPReport

Declaration
```
procedure SelectSection(const Name: WideString);
```

Description
Makes a certain section the current.

? IWPReport.SetAutomatic

Applies to
IWPReport

Declaration
```
function SetAutomatic(Mode: Integer; const Param: WideString): WordBool;
```

Description
Activate automatic mode. If that was not possible FALSE is returned. See propery RecordSet.

? IWPReport.SetProp

Applies to
IWPReport

Declaration
```
procedure SetProp(ID: Integer; const Value: WideString);
```

Description
Reserved

? IWPReport.ShowResult

Applies to
IWPReport

Declaration
```
procedure ShowResult;
```

Description
Displays the editor which holds the result of the reporting process.

? IWPReport.ShowTemplate

Applies to
IWPReport

Declaration
```
procedure ShowTemplate;
```

Description
Displays the editor which holds the template for the reporting process.

4.9.8.2 IWPReportBand

Manage report bands and groups

Description
This interface is used during report creation inside the OnReportState event.

Properties	Methods
Alias	**AddBand**
BandCount	**AddTable**

4.9.8.2 A) Properties

? Alias

Applies to
IWPReportBand

Declaration
```
string Alias;
```

Description
This parameter can be used for an alternative alias name.

? BandCount

Applies to
IWPReportBand

Declaration
```
int BandCount;
```

Description
Count of sub bands and groups. For sample code please see Report.Band(). The property is readonly.

? Count

Applies to
IWPReportBand

Declaration

```
int Count;
```

Description
This value is used during reporting. The current loop counter for each group is assigned to this property. The property is readonly.

? DBParams

Applies to
IWPReportBand

Declaration
```
property DBParams: WideString read Get_DBParams write Set_DBParams;
```

Description
This string parameter can be used freely, i.e. to store the SQL statement used for a sub query.

? Depth

Applies to
IWPReportBand

Declaration
```
int Depth;
```

Description
The nesting level of this band. The property is readonly.

? DisplayName

Applies to
IWPReportBand

Declaration
```
string DisplayName;
```

Description
This name should be displayed in the editor.

? GroupRadioNr

Applies to
IWPReportBand

Declaration
```
int GroupRadioNr;
```

Description
This property is used to create group of bands. They can be on different levels inside the report. When selection one of this band all other bands with same id will deselected - unless they use the same GroupSiblingNr as this band.

? GroupSiblingNr

Applies to
IWPReportBand

Declaration
```
int GroupSiblingNr;
```

Description
This property is used to create group of bands. If a band is selected all other bands which use the same GroupSiblingNr will be also selected.

? Mode

Applies to
IWPReportBand

Declaration
```
int Mode;
```

Description
Reserved Property.

? Name

Applies to
IWPReportBand

Declaration
```
string Name;
```

Description
The name of this group or band. Usually only group have names.

? Options

Applies to
IWPReportBand

Declaration
```
int Options;
```

Description
The following option bits are possible: Bit 1: Stretchable (reserved)
Bit 2: KeepControl (reserved)
Bit 3: NewPageAfter - create a new page after a group
Bit 4: NewPageBefore - create a new page before the group
Bit 5: DontContinueTable - do not continue a table which was started in a previous or nested group.

? ParAttr

Applies to
IWPReportBand

Declaration
`IWPAttrInterface ParAttr;`

Description
Currently not used.

? ParentGroupCount

Applies to
IWPReportBand

Declaration
```
int ParentGroupCount;
```

Description
This property contains the run count of the parent group of this band. It is only used during reporting. If undefined it is -1.

? ParentGroupName

Applies to
IWPReportBand

Declaration
```
string ParentGroupName;
```

Description
This is the name of the parent group.

? ParentParentGroup

Applies to
IWPReportBand

Declaration
```
string ParentParentGroup;
```

Description
This is the name of the parent group of the parent group.

? ParentParentGroupCount

Applies to
IWPReportBand

Declaration
```
int ParentParentGroupCount;
```

Description
This property contains the run count of the parent group of the parent group of this band. It is only used during reporting. If undefined it is -1.

? RecordSet

Applies to
IWPReportBand

Declaration
```
IUnknown RecordSet;
```

Description
You may assign an interface reference to access it inside the reporting events. This makes it easier to create multi level sub queries. TextDynamic will not use this interface.

? Selected

Applies to
IWPReportBand

Declaration
```
bool Selected;
```

Description
This property is used to change the selected state. If the Visibility bit 5 (=16) is set, a band will only be used if selected. Using the properties GroupRadioNr and GroupSiblingNr bands on different levels can be combined to logical groups.

? State

Applies to
IWPReportBand

Declaration
```
int State;
```

? Typ

Applies to
IWPReportBand

Declaration
```
int Typ;
```

? Visibility

Applies to
IWPReportBand

Declaration
```
int Visibility;
```

Description
For header and footer in first level (not children of a group) this property controls on which pages a page header or page footer should be created:
0=All pages, 1=Odd, 2=Even, 3=First, 4=AllNotFirst, 5=OddNotFirst, 6=Hide.

For group header and footer this value controls the usage of this band. The following bit are used: Bit 1: dont use at start or end of group. Bit 2: also use between data groups. Bit 5(=16): Only use if selected.

4.9.8.2 B) Methods

? IWPReportBand.AddBand

Applies to
IWPReportBand

Declaration
```
function AddBand(Typ: Integer; Visibility: Integer): Integer;
```

Description
If this is a group paragraph, AddPar will add a new child band. If this is a data, header or footer band the new band will be added after the text of this band.

Parameters	
Typ	Thy type for the new band, 0=data, 1=header, 2=footer and 3=group.
Visibility	For header and footer in first level (not children of a group) this controls on which pages a page header or page footer should be created: 0=All pages, 1=Odd, 2=Even, 3=First, 4=AllNotFirst, 5=OddNotFirst, 6=Hide. For group header and footer this value controls the usage of this band. The following bit are used: Bit 1: dont use at start or end of group. Bit 2: also use between data groups. Bit 5(value=16) can always be used. If this bet has been set the band can be selected and is only used if selected.

Returns
If no band was created 0 is returned, otherwise the ParPtr value of the created band paragraph.

? IWPReportBand.AddTable

Applies to
IWPReportBand

Declaration
```
procedure AddTable(ColCount: Integer; RowCount: Integer; Border: WordBool;
EventParam: Integer);
```

Description
This method creates a table inside group bands or after header, footer or data bands. If there is already a table all rows will be removed and the table will be recreated.

If the parameter EventParam is not 0 the event OnCreateNewCell will be triggerd for each new cell. You can use the event to create text and fields in the new cells.

Category
Table Support

? IWPReportBand.Band

Applies to
IWPReportBand

Declaration
```
function Band(Index: Integer): IWPReportBand;
```

Description
Retrieve sub bands. For sample code please see Report.Band().

? IWPReportBand.CheckSyntax

Applies to
IWPReportBand

Declaration
```
function CheckSyntax: Integer;
```

Description
This method is not used.

? IWPReportBand.Clear

Applies to
IWPReportBand

Declaration
```
procedure Clear(GroupsToo: WordBool);
```

Description
The methods delets the contents of this band. Groups will not be deleted unless the parameter GroupsToo was passed as true.

? IWPReportBand.FindVar

Applies to
IWPReportBand

Declaration
```
function FindVar(const Name: WideString): IWPReportVar;
```

Description
This function locates a group variable and returns null or the a reference to the interface IWPReportVar.

You can also use this function to **add** a variable, place in front of the name a '+' sign. The variable will be only added if it was not found. To delete a variable place in front the character '-'.

? IWPReportBand.GetParPtr

Applies to
IWPReportBand

Declaration
```
function GetParPtr: Integer;
```

Description
Get the ParPtr id of the paragraph which represents this band. You can use this id to move the cursor to this band.

? IWPReportBand.GetProp

Applies to
IWPReportBand

Declaration
```
function GetProp(Item: Integer): WideString;
```

Description
Reserved.

? IWPReportBand.GetVar

Applies to
IWPReportBand

Declaration
```
function GetVar(Index: Integer): IWPReportVar;
```

Description
This function returns an interface to read and modify a certain group variable. You can check all variables in a loop by using VarCount.

? IWPReportBand.ParentGroup

Applies to
IWPReportBand

Declaration
```
function ParentGroup: IWPReportBand;
```

Description
This functions returns either null or a reference of the parent group of this band.

? IWPReportBand.SetParPtr

Applies to
IWPReportBand

Declaration
```
procedure SetParPtr(Paragraph: Integer);
```

Description
This method may not be used.

? IWPReportBand.SetProp

Applies to
IWPReportBand

Declaration
```
procedure SetProp(Item: Integer; const Value: WideString);
```

Description
Reserved

? IWPReportBand.VarCount

Applies to
IWPReportBand

Declaration
```
function VarCount: Integer;
```

Description
This function returns the count of variables of a group.

4.9.8.3 IWPReportVar

Manage reporting variables (to sum values)

Description
This technology is under development. It is expected to be ready by Mid July 2006.

Properties	Methods
Description	**ElementsAdd**
Format	**ElementsClear**
GetTextFormula	**ElementsCount**
LoopFormula	**ElementsGet**
Mode	
Name	
ParStyle	
StartFormula	
Text	
Title	
Value	
WidthTW	

4.9.8.3 A) Properties

? Description

Applies to
IWPReportVar

Declaration

```
string Description;
```

Description
This is the description for this variable.

? Format

Applies to
IWPReportVar

Declaration
```
string Format;
```

Description
This is the format string. "CUR=$" will format as currency.

? GetTextFormula

Applies to
IWPReportVar

Declaration
```
string GetTextFormula;
```

? LoopFormula

Applies to
IWPReportVar

Declaration
```
string LoopFormula;
```

Description
This is the formula which will be evaluated before each time the group is processed. Use the string "=" to assign, or "+=", "-=", "*=" or "/=" in front of the formula to manipulate the current value.
In formulas you can refer to number data base fields by simply specify a name, i.e. ORDERS. PRICE. The event OnReadFormulaVar will be triggered to read the value. If the variable was not found the value will be always 0.

? Mode

Applies to
IWPReportVar

Declaration
```
int Mode;
```

? Name

Applies to
IWPReportVar

Declaration

```
string Name;
```

? ParStyle

Applies to
IWPReportVar

Declaration
```
string ParStyle;
```

? StartFormula

Applies to
IWPReportVar

Declaration
```
string StartFormula;
```

Description
This formula will be evaluated before the first time the parent group is used. You can use "=" in front of the formula to assign a value, othewise the value will be 0. Usuall you only have to use LoopFormula.

? Text

Applies to
IWPReportVar

Declaration
```
string Text;
```

Description
This is the value as text.

? Title

Applies to
IWPReportVar

Declaration
```
string Title;
```

Description
The title of this variable.

? Value

Applies to
IWPReportVar

Declaration
```
double Value;
```

Description

The value of this variable.

? WidthTW

Applies to
IWPReportVar

Declaration
```
int WidthTW;
```

Description
Reserved.

4.9.8.3 B) Methods

? IWPReportVar.ElementsAdd

Applies to
IWPReportVar

Declaration
```
procedure ElementsAdd(const Text: WideString; Value: Double; Color: Integer);
```

Description
Adds a string, a number and a color to this variable. The number is also added to the value. of the variable.
If the "CUR=" Format has been selected, the added value will be rounded to two decimal points before it is added.
Later a charting feature will be added which uses this feature. ElementsAdd can be also used in the LoopFormulas to create an element list while the report is beeing created.

? IWPReportVar.ElementsClear

Applies to
IWPReportVar

Declaration
```
procedure ElementsClear;
```

Description
Removes all elements and sets the value=0.

? IWPReportVar.ElementsCount

Applies to
IWPReportVar

Declaration
```
function ElementsCount: Integer;
```

Description
This is the number of elements.

? IWPReportVar.ElementsGet

Applies to
IWPReportVar

Declaration
```
function ElementsGet(Index: Integer; var Value: Double; var Color: Integer):
WideString;
```

Description
Retrieve a certain element.

4.11 PDF Creation, Email

4.11.1 IWPPdfCreator

Interface to start and configure PDF creation

Description
The integrated PDF engine is based on the powerful wPDF V3 library which is widely accepted by the industry.

It also is able to create PDF files which meet the PDF/A specification.
The PDF exporter is now now also able to attach data - see example.

Using SetProp it is now possible to only embed certain fonts as Type3 fonts, add info to the XMP of the PDF and also to set the copyright status of the PDF.

Properties:	**Methods**:
CIDFontMode	**BeginDoc**
Compression	**EndDoc**
FontMode	**GetProperty**
OwnerPW	**Print / Export** (Note: "Print" does not work in VB6 since it is predefined by IDE)
PDFAMode	**PrintSecond**
PDFFile	**SetProp** (set various properties)
Protection	**SetProperty**
Security	
UserWP	

4.11.1.1 Properties

4.11.1.1 A) CIDFontMode

Applies to
IWPPdfCreator

Declaration
```
int CIDFontMode;
```

Description
This property enables the support for text written used by Character Identifiers. Here in the PDF text numbers are written which are mapped to certain glyphs in an embedded font. A

special additional mapping table makes sure that the text can be extracted as unicode text. When the CID feature is used this means the fonts are embedded as subsets in a highly efficient way. PDF files become smaller this way. Since it works with character ids and not with charsets the export for say, Russian text, also works without having specified the charset explicitly. wPDF uses as character IDs the UNICODE values of each character - it can have used any number but we wanted to preserve the most information of the source text in the PDF as possible. Please note that the support for asian languages does NOT use the CID feature. Asian languages use special, predefined fonts and mapping tables and require the charset to be known to be properly exported.

CidFontMode Values:

0 - CID feature is not used. The property FontMode rules font embedding
1 - all fonts are embedded. Unicode values will be used as character ids
2 - only symbol fonts will be embedded

wPDF 4: You can now set FontMode to 7 instead of having to set CidFontMode to 1. This makes it easier to use a combobox to select the font mode since only one property has to be changed.

4.11.1.1 B) Compression

Applies to
IWPPdfCreator

Declaration
```
int Compression;
```

Description

Bit 1:
0: no compression
1: deflate compression for text

Bit 2-8:
0: no jpeg
1-100: JPEG compression for images

4.11.1.1 C) FontMode

Applies to
IWPPdfCreator

Declaration
```
int FontMode;
```

Description
Fontmode:
Use true type fonts but do not embed
0 : UseTrueTypeFonts

Use true type fonts and embed all
1 : EmbedTrueTypeFonts

Use true type fonts but only embed symbolfonts
2 : EmbedSymbolTrueTypeFonts

Dont use TTF fonts (not suggested)
3 : UseBase14Type1Fonts

Embed TTF fonts but only the current charset (255 char)
4 : EmbedSubsetTrueType_Charsets

Embed TTF fonts but only the actually used chars
5 : EmbedSubsetTrueType_UsedChar

Convert all fonts to Type3 and embed. Creates smallest files. [recommended, new in wPDF 4 / TextDynamic 7]
6: EmbedType3Fonts
To only embed certain fonts as Type3 fonts You can use SetProp(5, "\"Wingdings\", \"Webdings\"")

Create CID Subset Fonts (added to wPDF 4)
7: EmbedCIDFonts

Example:

```
rtF2PDF1.PdfCreator.FontMode = (int)wPDF.eFontMode.EmbedType3Fonts;
```

4.11.1.1 D) OwnerPW

Applies to
IWPPdfCreator

Declaration
```
string OwnerPW;
```

Description
The owner password for the PDF file. Also see UserPW.

4.11.1.1 E) PDFAMode

Applies to
IWPPdfCreator

Declaration
```
int PDFAMode;
```

Description
This property enables PDF/A support.

Values: 0=off or 1=enabled

PDF/A is a new norm which based on PDF 1.4 - it was created to provide a guildline for the creation of document files which stay readable for the time to come. So the major demand for PDF/A complient files is that the used font files are embedded. Security measures are forbidden in PDF/A complient files as are links to external files. But there is more to PDF/A. We have checked the component carefully against the final documentation of PDF/A.

When you use TextDynamic additional (invisible) tags will be added to the PDF data. This tags make it possible to identify layout elements (such as header or footer texts) on a page. They can be also used by a PDF reader to convert the PDF data into text paragraphs, something which is otherwise at least difficult and impossible if a paragraph spans a page. The wPDF

engine will also create tags to mark table cells. wPDF3 will also add the document information as XMP data to the PDF file.

4.11.1.1 F) PDFFile

Applies to
IWPPdfCreator

Declaration
```
string PDFFile;
```

Description
Default filename for the created PDF file. Unless you have licensed the TextDynamic server a file save dialog will be always displayed.

4.11.1.1 G) Protection

Applies to
IWPPdfCreator

Declaration
```
int Protection;
```

Description
PDF protection: bit 1: Enable protection
bit 2: Enable Printing
bit 3: Enable Changing
bit 4: Enable Copying
bit 5: Low Quality PrintOnly
bit 6: Enable DocAssembly
bit 7: Enable Form Field Fill In
bit 8: Enable Accessibility Options

4.11.1.1 H) Security

Applies to
IWPPdfCreator

Declaration
```
int Security;
```

Description
Security: 0=40 bit, 1=128 bit

4.11.1.1 I) UserWP

Applies to
IWPPdfCreator

Declaration
```
string UserWP;
```

Description
User Password for PDF file - it will be requested when the PDF file is opened. Also see OwnerPW .

Note - actually the name was intended to be UserPW :-)

4.11.1.2 Methods

4.11.1.2 A) IWPPdfCreator.BeginDoc

Applies to
IWPPdfCreator

Declaration
```
function BeginDoc: WordBool;
```

Description
Start a PDF file. This makes it possible to export several texts into one PDF file. You need to execute EndDoc to close the PDF file.

4.11.1.2 B) IWPPdfCreator.EndDoc

Applies to
IWPPdfCreator

Declaration
```
procedure EndDoc;
```

Description
Close a PDF file opened with BeginDoc.

4.11.1.2 C) IWPPdfCreator.GetProperty

Applies to
IWPPdfCreator

Declaration
```
function GetProperty(const Name: WideString): WideString;
```

Description
Read a PDF document property, such as "Author", "Keywords", "Subject".

4.11.1.2 D) IWPPdfCreator.Print, Export

Applies to
IWPPdfCreator

Declaration
```
function Print: WordBool;
function Export: WordBool;
```

Description
Export the text in editor #1.

VB6 Note:

If you are using VB6 you cannot use this method since Visual basic overrides this name with special functionality.

In TextDynamic we added the method "Export" to avoid this complication.

In wRTF2PDF You can use the command 1313 instead of PDFCreator.Print:

VB6 Example: wRTF2PDF:

```
Dim Memo As IWPEditor ' TextDynamic uses IWPMemo.
Dim PDFCreator As IWPPdfCreator
' TODO: Modify license codes
PDFControl1.StartEngine "wPDFControlDemo.dll", "lic", "lic", 0
Set Memo = PDFControl1.Memo
Set PDFCreator = PDFControl1.PDFCreator
Memo.LoadFromFile "sometext.rtf", False, "AUTO"
PDFCreator.PDFFile = "new.PDF"
' PDFCreator.Print - this cannot be used due to VB "bug"
' Use command instead
PDFControl1.ExecIntCommand 1313, 0
```

Category
Load and Save
Printing

4.11.1.2 E) IWPPdfCreator.PreparePDFAttachment

Applies to
IWPPdfCreator

Declaration

```
int PreparePDFAttachment(string filename; int Editor; int Modes): Integer;
```

Description
Write a PDF file with a temporary file and adds the name to the attachment list.

This function is intended to be used when sending e-mails.

The attachment list has to be prepared with Memo.PrepareAttachmentList.

You can select the editor #1 or #2 and also pass a file name which should be used. The parameter Modes is reserved.

The filename should not contain a path.

The created file will use part of the passed filename, but can have a different name, if a file already exists.

Please use GetAttachment to retrieve the attached file names.

When using TextDynamic, please note
- You need the **server license** - otherwise the PDF creator will always show a file open dialog!
Please remember to also activate the "Server" flag in the call to SetEditorMode

SetEditorMode 0, ... + 128, ... , 0

4.11.1.2 F) IWPPdfCreator.PrintSecond

Applies to
IWPPdfCreator

Declaration
```
function PrintSecond: WordBool;
```

Description
Export the text in editor #2.

4.11.1.2 G) IWPPdfCreator.SetProp

Applies to
IWPPdfCreator

Declaration
```
procedure SetProp(ID: Integer; const Value: WideString);
```

Description

The following values are possible for the parameter "ID".

0: Clears the value of the variable "lasterror". This variable can be read with GetProperty ("error")

1: Clears additional PDF info items

2: Clears the PDF excluded fontlist - this fonts are not embedded

3: Assigns the PDF excluded fontlist from a comma separated list

Added to wPDF V4 / TextDynamic 7:

Control Type3 embedding:

4: Clear the list EmbeddedType3Fonts. Fonts from this list as embedded as Type3 fonts, all other according to property FontMode.

5: Assign EmbeddedType3Fonts from a comma separated list

Add Copyright info the the PDF properties

6: Assign CopyrightNotice - this string will be added to the PDF XMP

7: Assign CopyrightURL - this string will be added to the PDF XMP

8: Assign CopyrightPublicDomain - Value may be "true" or "false"

Add blocks to the PDF XMP Section. This can be used to add ZUGFerD data to XMP. (ZUGFerD is a specification to support electronic invoices)

9: Assign XMPInfoExtension

10: Assign XMPInfoSchemaExtension

4.11.1.2 H) IWPPdfCreator.SetProperty

Applies to
IWPPdfCreator

Declaration

```
procedure SetProperty(const Name: WideString; const Value: WideString);
```

Description
Write a PDF document property, such as "Author", "Keywords", "Subject".

4.12 Helpers

4.12.1 IWPTextWriter

Access writing properties

Description
This interface is used by the event OnLoadExtImage. It provides access to few saving parameters: Format, SaveName and SavePath.

Properties	Methods
Format	WriteData
SaveName	WriteString
SavePath	

4.12.1.1 Properties

4.12.1.1 A) SaveName

Applies to
IWPTextWriter

Declaration
```
string SaveName;
```

Description
The name of the file which is beeing written.

4.12.1.1 B) SavePath

Applies to
IWPTextWriter

Declaration
```
string SavePath;
```

Description
The path of the file which is being written.

4.12.1.2 Methods

4.12.1.2 A) IWPTextWriter.WriteData

Applies to
IWPTextWriter

Declaration
```
procedure WriteData(Data: LongWord; DataLen: Integer);
```

Description

Write data from a buffer.

4.12.1.2 B) IWPTextWriter.WriteString

Applies to
IWPTextWriter

Declaration
```
procedure WriteString(const Text: WideString);
```

Description
Write a string to the output stream or file.

4.12.2 IWPPicture

.NET Image2Picture utility class

Description
This interface is used to allow the TextDynamic DLL to read data from a .NET image object. To use this interface simply create an instance of the class **Image2Picture** and pass it as Picture parameter to any of the methods which expects an IPicture interface.

Example:
```
WPDLLInt1.TextCursor.InputPicture(new WPDynamic.Image2Picture
(pictureBox1.Image),0,0);
```

4.12.3 IWPStream - Stream2WPStream

.NET Stream2WPStream utility class

Description
This interface is used to allow the TextDynamic DLL to read and write from and to .NET Streams. To use this interface simply create an instance of the class **Stream2WPStream** and pass it as Stream parameter to any of the methods which expects an IStream interface.

Load:

```
IWPMemo Memo = WPDLLInt1.Memo;
FileStream textstream = new FileStream("C:\\1.htm",FileMode.Open);
Memo.LoadFromStream(
 new WPDynamic.Stream2WPStream(textstream),
 false, "" );
textstream.Close();
```

Save:

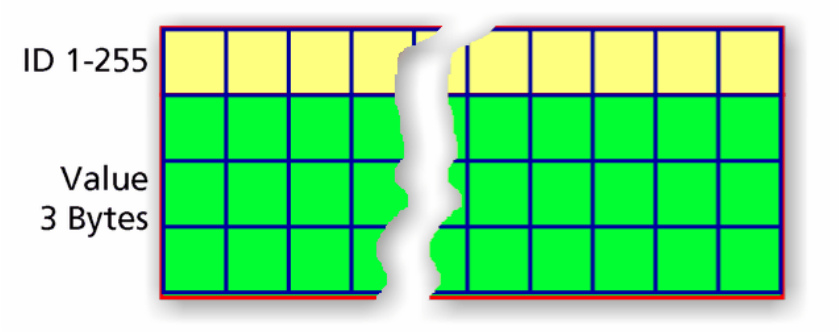
```
IWPMemo Memo = WPDLLInt1.Memo;
FileStream textstream = new FileStream("C:\\new.htm",FileMode.Create);
Memo.SaveToStream(
 new WPDynamic.Stream2WPStream(textstream),
 false, "HTML" );
textstream.Close();
```

4.13 WPAT_codes

TextDynamic stores paragraph and paragraph style attribute values (such as indents and spacing) using a value and an ID. The ID is the "WPAT code".

List of WPAT codes:
Character Attributes
Predefined Color Index Values
Paragraph Attributes
Numbering Attributes
Border Attributes
Table Size and Position

Technical Note:
Internally an array of 32bit values is used:



This makes it possible to not only store the value of a property but also to know that the property is not defined at all - but does not use memory for undefined properties.

The IDs are specified as WPAT_ constants. The first 16 are used for character attributes (and also used by the CharAttr records). The names of the procedures which work with this kind of attributes start with "A".

"WPAT" codes are used by low level methods to set and retrieve paragraph and style attributes.

You can use this properties whenever you manipulate paragraph, table cell or table row properties directly.

The Attributes do only use 3 bytes, so the maximum value is $8FFFFF.

The .NET assembly defines the enum WPAT, so you need to write (int)WPAT.CharFontSize to set the size of the text in pt*100.

Interface to use WPAT codes:
IWPParInterface.ParAAddBits
IWPParInterface.ParAClear
IWPParInterface.ParADel
IWPParInterface.ParADelBits
IWPParInterface.ParAGet
IWPParInterface.ParAInc
IWPParInterface.ParASet
IWPParInterface.ParStrCommand

4.13.1 Character Attributes

WPAT_CharFont = 1; // Set the index of the font.

The font index can be claculated using CurrPar.ParStrCommand(1,fontname,0). But usually you will not change the font index directly since you can use the method CurrAttr.SetFontFace to specify a font name.

WPAT_CharCharset = 2; // the CharSet of the font. Use IWPParInterface.ConvertFontnameToIndex to create an index value!

WPAT_CharFontSize = 3; // FontSize in pt*100
WPAT_CharWidth = 4; // Character scaling value (display similar to WPAT_CharSpacing)
WPAT_CharEffect = 5; // Special character effects and character styles
FLAG: WPEFF_CUSTOM1 = 1; // wpcustN - 63 Custom tags for whatever fixed styles you want to develop
FLAG: WPEFF_CUSTOMMASK = 63; // The following are bits
FLAG: WPEFF_SHADOW = 64; // \shad
FLAG: WPEFF_INSET = 128; // \embo
FLAG: WPEFF_OUTSET = 256; // \impr
FLAG: WPEFF_OUTLINE = 512; // \outl
FLAG: WPEFF_FRAME = 1024; // \chbrdr - only default \brdrs\brdrw10
FLAG: WPEFF_ANIMbit1 = 2048; // \animtext1
FLAG: WPEFF_ANIMbit2 = 4096; // \animtext2
FLAG: WPEFF_ANIMbit3 = 8192; // \animtext4
FLAG: WPEFF_ANIMMask = 8192 + 2048 + 4096;

Character styles are stored in 2 properties, one is the mask the other switches on and off:
WPAT_CharStyleMask = 6; // always used **together** with WPAT_CharStyleON to allow the combination of styles
WPAT_CharStyleON = 7; // Switch one or multiple of the following Styles on (WPSTY_BOLD ...)

FLAG: WPSTY_BOLD = 1; // Bit 1 bold
FLAG: WPSTY_ITALIC = 2; // Bit 2 italic
FLAG: WPSTY_UNDERLINE = 4; // Bit 3 underlined (solod)
FLAG: WPSTY_STRIKEOUT = 8; // Bit 4 strikeout
FLAG: WPSTY_SUPERSCRIPT = 16; // Bit 5 superscript
FLAG: WPSTY_SUBSCRIPT = 32; // Bit 6 subscript
FLAG: WPSTY_HIDDEN = 64; // Bit 7 hidden text
FLAG: WPSTY_UPPERCASE = 128; // Bit 8 all uppercase
FLAG: WPSTY_SMALLCAPS = 256; // Bit 9 all uppercase but non-captitals are 20% larger
FLAG: WPSTY_LOWERCASE = 512; // Bit 10 all lowercase
FLAG: WPSTY_NOPROOF = 1024; // Bit 11 \noproof - disable spellcheck for this
FLAG: WPSTY_DBLSTRIKEOUT = 2048; // Bit 12 strikeout - double solid line
FLAG: WPSTY_BUTTON = 4096; // button - not used!
FLAG: WPSTY_PROTECTED = 8192; // protected text - can be optionally handled as shaded text
FLAG: WPSTY_USERDEFINED = 16384; // Bit 15 user defined flag
{* bit 16 must be unused }

WPAT_CharColor = 8; // The text color (as index in palette - see Convert Utility)
Using ParStrCommand(3, color_string, 0) it is possible to convert a color name into an index value.
WPAT_CharBGColor = 9; // The text background color (as index in palette - see Convert Utility)
WPAT_CharSpacing = 10; // "Letter-Spacing" in twips, 0..$8000 = EXPAND, $8001- $FFFF = COMPRESS
WPAT_CharLevel = 11; // Move Character up or down - in half points (RTF: \up \dn }
// 0..$8000 = UP, $8001- $FFFF = down
WPAT_CharHighlight = 12; // {reserved} Highlight mode (different styles and colors)

WPAT_UnderlineMode = 13; // Underlining mode, 0=off, 1=solid, 2=double, 3= dotted ...
FLAG: WPUND_Standard = 1; // Underline Style 1
FLAG: WPUND_Dotted = 2;
FLAG: WPUND_Dashed = 3;
FLAG: WPUND_Dashdotted = 4;
FLAG: WPUND_Dashdotdotted = 5;
FLAG: WPUND_Double = 6;
FLAG: WPUND_Heavywave = 7;
FLAG: WPUND_Longdashed = 8;
FLAG: WPUND_Thick = 9;
FLAG: WPUND_Thickdotted = 10;
FLAG: WPUND_Thickdashed = 11;
FLAG: WPUND_Thickdashdotted = 12;
FLAG: WPUND_Thickdashdotdotted = 13;
FLAG: WPUND_Thicklongdashed = 14;
FLAG: WPUND_Doublewave = 15;
FLAG: WPUND_WordUnderline = 16;
FLAG: WPUND_wave = 17;
FLAG: WPUND_curlyunderline = 18; // only used for spellcheck
FLAG: WPUND_NoLine = 200; // Dont draw line !!!! When imported from RTF!
WPAT_UnderlineColor = 14; // Underlining color, 0=text color, otherwise colorindex +1 - - see Convert Utility
WPAT_TextLanguage = 15; // {reserved} Language of the text
WPAT_CharStyleSheet = 16; // CharacterStyle (index in ParStyles)

NOTE: gray symbols are reserved for future versions of TextDynamic!

4.13.2 Predefined Color Index Values

The methods CurrAttr.ColorToNr and CurrAttr.NrToColor are used to convert between index values and RGB values.

This index values are always defined:

```
0=Black (Default    9=Aqua
1=Red               10=Teal
2=Green             11=Navy
3=Blue              12=White
4=Yellow,           13=Light Gray
5=Fuchsia           14=Gray
6=Purple            15=Black
7=Maroon
8=Lime
```

Also see the convert utility functions in IWPParInterface.

4.13.3 Paragraph Attributes

// Margins
WPAT_IndentLeft = 17 - the left indent in twips
WPAT_IndentRight = 18; - the right indent in twips
WPAT_IndentFirst = 19; - the first indent in twips (can be negative, too)
WPAT_SpaceBefore = 20 - the space before a paragraph
WPAT_SpaceAfter = 21; - the space after a paragraph
WPAT_LineHeight = 22; - the LineHeight in in % (Has priority over WPAT_SpaceBetween)
WPAT_SpaceBetween = 23; - the space between paragraphs. When negative : Absolute, When Positive minimum

// padding, only in tables:
WPAT_PaddingLeft = 24 - Distance from Border to Text in twips (CSS = padding) tscellpaddt / trpaddl
WPAT_PaddingRight = 25 - Distance from Border to Text (CSS = padding)
WPAT_PaddingTop = 26- Distance from Border to Text (CSS = padding)
WPAT_PaddingBottom = 27- Distance from Border to Text (CSS = padding)

// Alignment
WPAT_Alignment = 29 - horizontal alignment: 0=left, 1=center, 2=right, 3=justified
WPAT_VertAlignment = 30 - vertical alignment: 0=top, 1=center, 2=bottom

// Colors and background
WPAT_BGColor = 50; // Background Color - (as index value) - see Convert Utility
WPAT_FGColor = 51; // Foreground Shading Color - - see Convert Utility
WPAT_ShadingValue = 52; // Background Shading Percentage in %
WPAT_ShadingType = 53; // Background Shading Type
VALUE: WPSHAD_solidbg = 0; // Solid Background - use WPAT_BGColor and WPAT_ShadingValue
VALUE: WPSHAD_solidfg = 1; // Solid Background - use WPAT_FGColor and WPAT_ShadingValue
VALUE: WPSHAD_clear = 2; // Clear Background
VALUE: WPSHAD_bdiag = 3; // Backward diagonal pattern.
VALUE: WPSHAD_cross = 4; // Cross pattern.
VALUE: WPSHAD_dcross = 5; // Diagonal cross pattern.
VALUE: WPSHAD_dkbdiag = 6; // Dark backward diagonal pattern.
VALUE: WPSHAD_dkcross = 7; // Dark cross pattern.
VALUE: WPSHAD_dkdcross = 8; // Dark diagonal cross pattern.
VALUE: WPSHAD_dkfdiag = 9; // Dark forward diagonal pattern.
VALUE: WPSHAD_dkhor = 10; // Dark horizontal pattern.
VALUE: WPSHAD_dkvert = 11; // Dark vertical pattern.
VALUE: WPSHAD_fdiag = 12; // Forward diagonal pattern.
VALUE: WPSHAD_horiz = 13; // Horizontal pattern.
VALUE: WPSHAD_vert = 14; // Vertical pattern.
WPAT_BGBitMap = 54; // Background Image.
WPAT_BGBitMapMode = 55; // Bitfield for scroll, center, center, repeat

Special Flags:
WPAT_ParProtected = 92; // 1=protect, 0=not protected
WPAT_ParKeep = 93; // 1=no page break
WPAT_ParKeepN = 94; // 1=keep paragraphs together

WPAT_ParIsOutline = 160; // Mark as outline (used by PDF Export to create "Bookmarks" and for TOC). The value is the level. 0 = not marked. You can use Memo.TextCommand(7) to set this flag according to the used style names. (Use TextCommand(3) to create a TOC.)

NOTE: gray symbols are reserved for future versions of TextDynamic!

4.13.4 Numbering Attributes

WPAT_NumberSTYLE = 31; // Reference to a different Style. It can be a single style or
// a style which is part of an outline style sheet. In the letter case the correct
// style will be be located inside this group using the WPAT_NumberLEVEL parameter.

WPAT_NumberLEVEL = 32; // Numbering Level
// The following styles are used for simple paragraph numbering and also inside
// numbering styles. Numbering styles can also use all other paragraph attributes!
// Please also note that 'WPAT_NumberLEVEL' on its own does NOT activate numbering.

```
  // WPAT_NumberSTYLE or WPAT_NumberMODE must be also specified.
  // If only WPAT_NumberLEVEL is used this just specifies the current outline level.

  WPAT_OUTLINELEVEL = 32; // synonym for WPAT_NumberLEVEL
  WPAT_NumberMODE = 33; // Supported elements are: (\levelnfcN)
  {*}WPNUM_ARABIC = 1; // Arabic (1, 2, 3)
  {*}WPNUM_UP_ROMAN = 2; // Uppercase Roman numeral (I, II, III)
  {*}WPNUM_LO_ROMAN = 3; // Lowercase Roman numeral (i, ii, iii)
  {*}WPNUM_UP_LETTER = 4; // Uppercase letter (A, B, C)
  {*}WPNUM_LO_LETTER = 5; // Lowercase letter (a, b, c)
  {*}WPNUM_LO_ORDINAL = 6; // Lowercase Ordinal number (1st, 2nd, 3rd)
  {*}WPNUM_Text = 7; // Cardinal text number (One, Two Three)
  {*}WPNUM_ORDINAL_TEXT = 8; // Ordinal text number (First, Second, Third)
  {*}WPNUM_WIDECHAR = 15; // Double-byte character
  {*}WPNUM_CHAR = 16; // Single-byte character
  {*}WPNUM_CIRCLE = 19; // Circle numbering (*circlenum)
  {*}WPNUM_ARABIC0 = 23; // Arabic with leading zero (01, 02, 03, ..., 10, 11)
  {*}WPNUM_BULLET = 24; // Bullet (no number at all)

  WPAT_NumberTEXTB = 34; // Text Before (Use TextToNumber to create an index from a
string)
  WPAT_NumberTEXTA = 35; // Text After
  WPAT_NumberTEXT = 36; // Char #1..#10 are the level placeholders, the rest is the
surrounding text
  WPAT_Number_STARTAT = 37; // Start Number, also See WPAT.NumberStart
  WPAT_Number_ALIGN = 38; // 0=Left, 1=Center, 2=Right
  WPAT_Number_SPACE = 39; // Minimum distance from the right edge of the number to the
start of the paragraph text
  WPAT_NumberFONT = 40; // Optional: Font for the text
  WPAT_NumberFONTSIZE = 41; // Optional: FontSize  (pnfs) in pt * 100
  WPAT_NumberFONTCOLOR = 42; // Optional: FontColor (pncf)
  WPAT_NumberFONTSTYLES = 43; // Optional: CharStyles bitfield
  WPAT_NumberINDENT = 44; // Old Style Indent - NumberStyles may also use the
INDENTFIRST/INDENTLEFT props!
  WPAT_NumberFLAGS = 45; //
  {*}WPNUM_FLAGS_COMPAT = 1; // Compatibility to old RTF
  {*}WPNUM_FLAGS_USEPREV = 2; // Use text from previous level ( pnprev)
  {*}WPNUM_FLAGS_USEINDENT = 4; // Use Indent from previous level
  {*}WPNUM_FLAGS_FOLLOW_SPACE = 8; // A space follows the number, Default = TAB!
  {*}WPNUM_FLAGS_FOLLOW_NOTHING = 16; // nothing follows the number
  {*}WPNUM_FLAGS_LEGAL = 32; // convert previous levels to arabic
  {*}WPNUM_FLAGS_NORESTART = 64;
    // if this level does not restart its count each time a number of a higher level is reached
  {*}WPNUM_FLAGS_ONCE = 128; // Number each cell only once in a table
  {*}WPNUM_FLAGS_ACROSS = 256; // Number across rows (the default is to number down
columns)
  {*}WPNUM_FLAGS_HANG = 512; // Paragraph uses a hanging indent
  {*}WPNUM_NONumberING = 1024; // DO NOT NumberATE!
  {*}WPNUM_NumberSKIP = 2048; // Increase number but do not display
  WPAT_NumberPICTURE = 46; // Reserved for number pictures
  WPAT_Number_RES1 = 47;
  WPAT_Number_RES2 = 48;
  WPAT_Number_RES3 = 49;

  WPAT_NumberStart = 159; // a certain start number for numbering (used in paragraphs, do
not mix up with WPAT_Number_STARTAT)
  WPAT_ParIsOutline = 160; // Is Outline (used by PDF Export) - value = level!
```

4.13.5 Border Attributes

The WPAT_ codes used for border parameters are:

```
WPAT_BorderTypeL = 60; // Border Mode Left(no, single, double etc)
WPAT_BorderTypeT = 61; // Border Mode Top (no, single, double etc)
WPAT_BorderTypeR = 62; // Border Mode Right(no, single, double etc)
WPAT_BorderTypeB = 63; // Border Mode Bottom (no, single, double etc)
WPAT_BorderTypeDiaTLBR = 64; // Diagonal Line - TopLeft/BottomRight ( \cldglu )
WPAT_BorderTypeDiaTRBL = 65; // Diagonal Line - TopRight/BottomLeft
```

The following constants are used as values for the 'Type' attribute

```
{*}WPBRD_SINGLE = 0; // \brdrs  Single-thickness border. (=Default)
{*}WPBRD_NONE = 1; // \brdrnil \brdrtbl  No Border
{*}WPBRD_DOUBLEW = 2; // \brdrth Double-thickness border.
{*}WPBRD_SHADOW = 3; // \brdrsh Shadowed border.
{*}WPBRD_DOUBLE = 4; // \brdrdb Double border.
{*}WPBRD_DOTTED = 5; // \brdrdot Dotted border.
{*}WPBRD_DASHED = 6; // \brdrdash      Dashed border.
{*}WPBRD_HAIRLINE = 7; // \brdrhair    Hairline border.
{*}WPBRD_INSET = 8; // \brdrinset      Inset border.
{*}WPBRD_DASHEDS = 9; // \brdrdashsm   Dashed border (small).
{*}WPBRD_DOTDASH = 10; // \brdrdashd   Dot-dashed border.
{*}WPBRD_DOTDOTDASH = 11; // \brdrdashdd      Dot-dot-dashed border.
{*}WPBRD_OUTSET = 12; // \brdroutset   Outset border.
{*}WPBRD_TRIPPLE = 13; // \brdrtriple  Triple border.
{*}WPBRD_THIKTHINS = 14; // \brdrtnthsg Thick-thin border (small).
{*}WPBRD_THINTHICKS = 15; // \brdrthtnsg      Thin-thick border (small).
{*}WPBRD_THINTHICKTHINS = 16; // \brdrtnthtnsg       Thin-thick thin border (small).
{*}WPBRD_THICKTHIN = 17; // \brdrtnthmg Thick-thin border (medium).
{*}WPBRD_THINTHIK = 18; // \brdrthtnmg Thin-thick border (medium).
{*}WPBRD_THINTHICKTHIN = 19; // \brdrtnthtnmg Thin-thick thin border (medium).
{*}WPBRD_THICKTHINL = 20; // \brdrtnthlg      Thick-thin border (large).
{*}WPBRD_THINTHICKL = 21; // \brdrthtnlg      Thin-thick border (large).
{*}WPBRD_THINTHICKTHINL = 22; // \brdrtnthtnlg       Thin-thick-thin border (large).
{*}WPBRD_WAVY = 23; // \brdrwavy Wavy border.
{*}WPBRD_DBLWAVY = 24; // \brdrwavydb  Double wavy border.
{*}WPBRD_STRIPED = 25; // \brdrdashdotstr     Striped border.
{*}WPBRD_EMBOSSED = 26; // \brdremboss Embossed border.  (CSS=ridge)
{*}WPBRD_ENGRAVE = 27; // \brdrengrave Engraved border.  (CSS=groove)
{*}WPBRD_FRAME = 28; // \brdrframe     Border resembles a "Frame."
```

The following properties store the width in twips:

```
WPAT_BorderWidthL = 66; // Thickness left inner Line
WPAT_BorderWidthT = 67; // Thickness top inner Line
WPAT_BorderWidthR = 68; // Thickness right inner Line
WPAT_BorderWidthB = 69; // Thickness bottom inner Line
WPAT_BorderWidthDiaTLBR = 70; // Diagonal Line - TopLeft/BottomRight
WPAT_BorderWidthDiaTRBL = 71; // Diagonal Line - TopRight/BottomLeft
```

These values store the color of the borders:

```
WPAT_BorderColorL = 72; // Color left inner Line - this is an index value. You need to
use the Convert function
WPAT_BorderColorT = 73; // Color top inner Line
WPAT_BorderColorR = 74; // Color right inner Line
WPAT_BorderColorB = 75; // Color bottom inner Line
```

```
  WPAT_BorderColorDiaTLBR = 76; // Diagonal Line - TopLeft/BottomRight
  WPAT_BorderColorDiaTRBL = 77; // Diagonal Line - TopRight/BottomLeft
```

If the values of all borders are the same, these codes can be used as a shortcut to set the value for a whole group:

```
  // Shortcut - set width and color of ALL lines. - Use Flags to switch on/off
  WPAT_BorderType = 87; // Border Mode ALL LINES AROUND BOX
  WPAT_BorderWidth = 88; // Thickness ALL LINES
  WPAT_BorderColor = 89; // Color ALL LINES - this is an index value. You need to use the
Convert function
```

This is the most important code. By setting a single bit a border is switched on.

```
  // Border Flags - switch borders on/off
  WPAT_BorderFlags = 90;
  {The following flags switch on certain borders. The type can be defined or inherited}
  {*}WPBRD_DRAW_Left = 1; // Left Border (Default = Single Line = Mode 0) = BlLeft
  {*}WPBRD_DRAW_Top = 2; // Top Border
  {*}WPBRD_DRAW_Right = 4; // Right Border
  {*}WPBRD_DRAW_Bottom = 8; // Bottom Border
  {*}WPBRD_DRAW_All4   = 15; // left + Top+ Right + Bottom
  {*}WPBRD_DRAW_DiagLB = 16; // Cells: Diagonal Top-Left -> Bottom-Right
  {*}WPBRD_DRAW_DiagRB = 32; // Cells: Diagonal Top-Right-> Bottom-Left
  {*}WPBRD_DRAW_Bar = 64; // Border outside (right side of odd-numbered pages,
  //  left side of even-numbered pages)
  {*}WPBRD_DRAW_InsideV = 128; // Rows: Inside Vertical
  {*}WPBRD_DRAW_InsideH = 256; // Rows: Inside Horizontal
  {*}WPBRD_DRAW_Box = 512; // Draw Box around paragraph
  {*}WPBRD_DRAW_Finish = 1024; // Draw bottom border here, even if next paragraph has same
border properties
```

4.13.6 Table Size and Position

A table usually uses the complete text area, this is the page width minus the left and right margins.

Property to set the width in twips: WPAT_BoxWidth

Property to set the width in percent * 100 of the page width
WPAT_BoxWidth_PC

Offset from the left margin. Can be negative to be to the lefgt of the text area.
WPAT_BoxMarginLeft

Offset from the left margin. If negative the table is extended into the margin area
WPAT_BoxMarginRight

Horizontal Alignment of the table: WPAT_Box_Align:
  possible values are WPBOXALIGN_RIGHT and WPBOXALIGN_HCENTERTEXT

The column width is defined by the properties **WPAT_ColWidth** and WPAT_ColWidth_PC. The first is the width in twips, the latter is the width in percent * 10 (Example: 1000 sets a width of

10%).

WPAT_ColWidth has priority over WPAT_ColWidth_PC. You will need to delete the WPAT_ColWidth flag if you need to set the width in percent.

```
CurrPar.ParADel(WPAT_ColWidth)
CurrPar.ParASet(WPAT_ColWidth_PC, 1000); // 10%
```

You can read the current width of a cell using the method cell.IsWidthTW. The text must be formatted. Once the user resizes columns in a table all widths which are defined by a percent value will be converted into exact WPAT_ColWidth values. You can force this conversion for the complete text using the function WPRichText.TableFixAllCellWidths. This method can also round the width values suing a 'snap value'. The method TableAdjustCellWidth can be used to recalculate the width of all columns to keep them visible.

The property WPAT_BoxMinHeight can be used with table rows to set the minimum height of a table row.
WPAT_BoxMaxHeight is used to set the maximum width. Both properties will be ignored in cells which are merged vertically. They can be used together when using WPT format, in RTF only the minimum value can be preserved if both are used.

Note: There are some commands available to read and set the **current row height** using the method ParCommand.

5 Tasks (Problems and their Solution)

5.1 RTF2PDF in ASP.NET

1) In your ASP.NET C# project please **first add a reference** to the wPDF4.DLL. This DLL is a wrapper for the engine. The full version includes the source code for this wrapper, it is written in C#. You may need to recompile it for the .NET frame work you use.

2) In the file module you want to use RTF2PDF add a link to wPDF:
using wPDF;
using WPDynamic;

3) Within the Page_Load event you create and instance of RTF2PDF - at the end don't forget Dispose()!

```
RTF2PDF wpdllint1 = new RTF2PDF();
try
{
// insert the following code here
}
finally
{
  wpdllint1.Dispose();
}
```

4) Please set the license code. In ASP project please load the license from an encrypted file - this avoids that the keys become public in case the asp code is displayed by the server in case of an error. You can use "Demo.EXE" to create such an encrypted file. We place the license file outside of wwwroot, so it is not possible to access it.

```
wpdllint1.SetLicense("@FILE@mysecretpassword", "C:\\Windows\\rtflic.dat",0);
```

5) You can use code to load data, or to create text. Here we simply create a table.

```
wpdllint1.Memo.Clear(false,false);
wpdllint1.TextCursor.AddTable("",2,30,true,0,true,true);
```

6) When the document was created you can send it as response either as HTML or as PDF:

a) as HTML:

```
Response.Write( wpdllint1.Memo.SaveToString(false, "HTML") );
```

b) as PDF:

```
// Create PDF in memory
wpdllint1.PdfCreator.PDFFile = "memory";
wpdllint1.PdfCreator.FontMode = 0;
wpdllint1.Memo.ReformatAll(false,false);
wpdllint1.Print();

// Set ASP result
Response.Clear();
Response.ContentType = "application/pdf";
Response.AddHeader("Content-Type", "application/pdf");
Response.AddHeader("Content-Disposition","inline;filename=PortableDocument.pdf");
Response.BinaryWrite(wpdllint1.ResultBuffer);
```

Tip: In our demos we use this code to either create PDF or HTML output:

```
if (Request.QueryString.Get("pdf")!="true")
  // method a)
else
  // method b)
```

You can also create the output as RTF text:

```
if (Request.QueryString.Get("rtf")=="true")
{
  Response.Clear();
  // Add new header  for RTF
  Response.ContentType = "application/rtf";
  Response.AddHeader("Content-Type", "application/rtf");
  Response.AddHeader("Content-Disposition","inline;filename=" + afile + ".rtf");
            object buf = wpdllint1.Memo.SaveToVar(false, "RTF");
            if(buf!=null) Response.BinaryWrite((byte[])buf);
  Response.BinaryWrite(buf);
}
```

5.2 ASP .NET Troubleshooting

1) The PDF creation will not run on a server if the debug mode was activated. The debugmode created EMF files during the creation of a PDF and will not work on a webserver due to write restrictions.
So please disable wpdllint1.DebugPath = Server.MapPath(".");

2) If the webserver creates a message that a DLL could not be loaded, please check the 32 bit vs 64 bit mode. If your program was compiled for 32 bit, you need to create an "Application

Pool", enable 32bit in its extended properties and assign that pool to your web application (which must be also added) in the IIS manager.

3) The license file cannot be loaded from C:\Windows. Please use a different path for the license file which is not in the www root, i.e. wpdllint1.SetLicense("@FILE@" + password, "C:\ \License\\rtflic.dat", 0);
Please make sure the the user IIS_IUSRS have read access to the directory, otherwise the license file cannot be read.

4) The program works locally, but not on the webserver. Please check
a) read access of IIS_IUSRS to the license file (see 4.)
b) 32 bit mode enabled for application pool
c) web server uses same or higher .NET framework version.
d) The asp DLLs must be in subfolder /bin
e) the RTF2PDF or PDFControl DLLs were also copied

Visual Studio starts a special webserver for debuging.
This webserver takes it much easier with access rights. You can enable IIS usage in the debugging options.

5.3 ASP TransferHelper (RTF2PDF)

If you create the PDF in memory (as described here) your web service can be negatively affected in case the web user simply clicks the reload button several times. We created a download class which handles this case better. It first creates a temporary file and then writes small chunks of this file to the Response - until it detects that the client is not connected anymore or all data has been sent.
As advantage over TransmitFile() we here have the possibility to react on an aborted transmission and delete the temporary file when it is not needed anymore. Also see our test server at http://www.rtf-net.com.

```
///------------------------------------------------------------------------
/// The TransferHelper class is used inside the PageLoad event. Simply provide
/// the Response and the RTF2PDF object to the constructor.
/// Call Send( format, filename ) to send the data. The following format names
/// are supported: rtf, htm, pdf and wpt
///------------------------------------------------------------------------

public class TransferHelper : IDisposable
{
   // We need this to prepare the response
   private HttpResponse Response;
   // We work with this engine
   private RTF2PDF pdfcontrol;
   // To transfer we are using a temporary file - this is the generic ".TMP" placeholde
   private string TempFile = "";
   // and this the file with content and extension
   private string TempFileSend = "";
   // Constructor, simply assign the engine and response
   public  TransferHelper(HttpResponse aResponse, RTF2PDF aRTF2PDF)
   {
```

```
      Response  =  aResponse;
      pdfcontrol = aRTF2PDF;
   }
   // Dispose - delete the temporary files
   public void Dispose()
   {
      if (TempFileSend!="")
      { System.IO.File.Delete(TempFileSend);
         TempFileSend = "";
      }
      if (TempFile!="")
      { System.IO.File.Delete(TempFile);
         TempFile = "";
      }
   }
   // ------------------------------------------------------------
   // Now send a file - this is called from within PageLoad
   // ------------------------------------------------------------
   public bool Send(string format, string filename)
   {
      // Client is not there or file was already sent
      if (!Response.IsClientConnected)
      {
         Response.Close();
         return true; // ok
      }
      // We need a temporary file
      if (TempFile=="") TempFile = System.IO.Path.GetTempFileName();
      bool ok = true;
      // RTF Format
      if (format=="rtf")
      {
         TempFileSend = System.IO.Path.ChangeExtension(TempFile, ".RTF");
         // Tables must be measured to be exported to Word
         pdfcontrol.Memo.ReformatAll(false,false);
         if (!Response.IsClientConnected)
         TempFileSend = "";
         else if(!pdfcontrol.Memo.SaveToFile(TempFileSend, false, ""))
         {
            System.IO.File.Delete(TempFileSend);
            TempFileSend = "";
            ok = false;
         }
         else
         {
            Response.Clear();
            Response.ContentType = "application/rtf";
            Response.AddHeader("Content-Type", "application/rtf");
            Response.AddHeader("Content-Disposition","inline;filename="
                                    + filename + ".rtf");
         }
      }
      // HTML Format
      else if (format=="htm")
      {
         TempFileSend = System.IO.Path.ChangeExtension(TempFile, ".HTM");
         if(!pdfcontrol.Memo.SaveToFile(TempFileSend, false, ""))
         {
```

```
            System.IO.File.Delete(TempFileSend);
            TempFileSend = "";
            ok = false;
        }
    }
    // WPT (WPTools) Format
    else if (format=="wpt")
    {
        TempFileSend = System.IO.Path.ChangeExtension(TempFile, ".WPT");
        if(!pdfcontrol.Memo.SaveToFile(TempFileSend, false, ""))
        {
            System.IO.File.Delete(TempFileSend);
            TempFileSend = "";
            ok = false;
        }
        else
        {
            Response.Clear();
            Response.ContentType = "application/wpt";
            Response.AddHeader("Content-Type", "application/wpt");
            Response.AddHeader("Content-Disposition","inline;filename="
                                    + filename + ".wpt");
        }
    }
    // PDF Format
    else if (format=="pdf")
    {
        TempFileSend = System.IO.Path.ChangeExtension(TempFile, ".PDF");
        pdfcontrol.PdfCreator.PDFFile = TempFileSend;
        pdfcontrol.PdfCreator.FontMode = 0;
        pdfcontrol.Memo.ReformatAll(false,false);
        if (!Response.IsClientConnected)
        TempFileSend = "";
        else   if(!pdfcontrol.PdfCreator.Print())
        {
            System.IO.File.Delete(TempFileSend);
            TempFileSend = "";
            ok = false;
        }
        else
        {
            Response.Clear();
            Response.ContentType = "application/pdf";
            Response.AddHeader("Content-Type", "application/pdf");
            Response.AddHeader("Content-Disposition","inline;filename="
                                     + filename + ".pdf");
        }
    }
    // Unknown format
    else
    {
        TempFileSend = "";
        ok = false;
    }
    // --------- now send and while doing so check IsClientConnected ----------
    if (TempFileSend!="")
    {
        const int PartSize = 8192;
```

```
            byte[] buffer = new byte[PartSize];
            System.IO.FileStream Stream =
            new System.IO.FileStream(TempFileSend, System.IO.FileMode.Open);
            using(Stream)
            {
               int l;
               long len = Stream.Length;
               while ((len>0)&& Response.IsClientConnected)
               {
                  l = Stream.Read(buffer, 0, PartSize);
                  Response.OutputStream.Write(buffer, 0, l);
                  Response.Flush();
                  len -= l;
               }
            }
            Response.Close();
         }
         return ok;
      }
        }
```

You can call this method at the end of your PageLoad handler.

This demo simply takes a file (must be in directory c:\doc) and converts to PDF:

```
RTF2PDF wpdllint1 = new RTF2PDF();
try
{
   string path = Server.MapPath(".") + "\\";
   wpdllint1.SetLicense("@FILE@"+password, "C:\\Windows\\rtflic.dat",0);
   WPDynamic.IWPEditor Memo = wpdllint1.Memo;
   WPDynamic.IWPTextCursor TextCursor = Memo.TextCursor;
   //-----------------------------------------------------------------------
   string afile = System.IO.Path.GetFileName(Request.QueryString.Get("file"));
   string afilepath = "c:\\doc\\" + afile;

   // Try to load - if fails use "Memo" to display the error message
   if (!Memo.LoadFromFile(afilepath,false,""))
   {
      Memo.Clear(false, false);
      TextCursor.CPPosition = 0;
      TextCursor.InputText("Cannot open file \"" + afile +"\"");
      Response.Write( wpdllint1.Memo.SaveToString(false, "HTML") );
   }
   else
   {
      // Optional: Add a footer  ---------------------------------------------
      // Insert right tab for page numbering ---------------------------------
      TextCursor.InputFooter(0,"","");
      TextCursor.InputTabstop(true, (int)(29.7/2.54*1440), 1,0);
      TextCursor.InputText("Page ");
      TextCursor.InputFieldObject("PAGE","","1");
      TextCursor.InputText("/");
      TextCursor.InputFieldObject("NUMPAGES","","1");
      TextCursor.GotoBody();
      // and use our helper class to export as PDF ---------------------------
      TransferHelper transfer = new TransferHelper(Response, wpdllint1);
      using(transfer)
```

```
      {
         transfer.Send("pdf",afile);
      }
   }
}
finally
{
   wpdllint1.Dispose();
}
```

5.4 Load & Save

Interface "Memo" includes methods for loading and saving data quietly:

```
bool LoadFromFile(string filename, bool Insert, string FormatStr);
bool SaveToFile(string filename, bool OnlySelection, string FormatStr);
bool LoadFromStream(object Stream, bool Insert, string FormatStr);
bool SaveToStream(object Stream, bool OnlySelection,string FormatStr);
bool LoadFromString(string Data, bool Insert, string FormatStr);
string SaveToString(bool OnlySelection, string FormatStr);
```

If the parameter "Insert" is true the text will be inserted at the current cursor position.

If the parameter "OnlySelection" is true only the selected text will be saved.

The parameter FormatStr controls the format to create or load.

Possible values are "AUTO" (default), "RTF", "ANSI", "HTML", "XML", "XMLTAGS" and "UNICODE".
When reading text the format is detected automatically if an empty string or "AUTO" was specified.

"RTF" selects the RTF reader and writer. RTF is a text format standard which is widely understood.
"HTML" selects the HTML/CSS reader or writer
"XML" writes simplified HTML code. You can use "XML-useptag,-ignorefonts" to only write <p>, <b>, <i>, <u> and <a> tags. Paragraphs may use styles which are written as class="".
"XMLTAGS" is used to select a special XML reader which creates tag objects. (see: SetXMLSchema)
"ANSI" writes standard ANSI text. You can use "ANSI-codepageXXX" to write text for a certain code page
"UNICODE" reads and writes unicode (2 byte) data

Additional parameters are possible, such as "RTF-onlybody". (See formatstrings.htm)

The "Stream" parameter for LoadFromStream is an IStream or IWPStream interface. An IWPStream interface is provided by the utility class Stream2WPStream which is implemented by the c# wrapper.

This example loads from a file stream - the format is automatically detected.

```
      IWPMemo Memo = WPDLLInt1.Memo;
      FileStream textstream = new FileStream("C:\\1.htm",FileMode.
Open);
      Memo.LoadFromStream(
               new WPDynamic.Stream2WPStream(textstream),
               false, "" );
      textstream.Close();
```

This example creates HTML code in a new file stream:

```
      IWPMemo Memo = WPDLLInt1.Memo;
      FileStream textstream = new FileStream("C:\\new.htm",FileMode.
Create);
      Memo.SaveToStream(
               new WPDynamic.Stream2WPStream(textstream),
               false, "HTML" );
      textstream.Close();
```

When you work with a **data bound editor** you need to use TextCursor.CheckState(10) to trigger the PropChanged before any code which updates the text.

5.5 Create Text using code

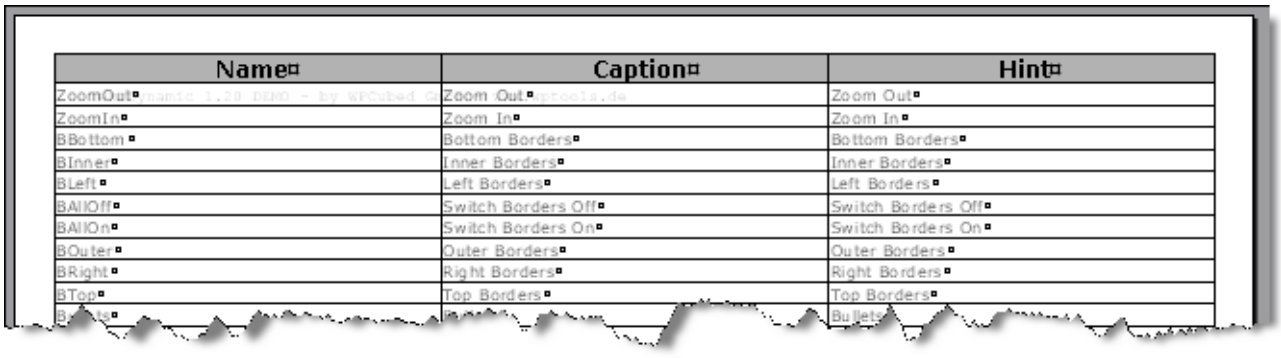
We want to show all the wpa actions nicely formatted in the editor.

To the event handler of a newly created button we add this code to create a table in the editor.

```
   // Now clear the text
   WPDLLInt1.Clear();

   // We need these interfaces for text creation
      IWPMemo Memo = WPDLLInt1.Memo;
      IWPTextCursor TextCursor = Memo.TextCursor;
      IWPAttrInterface AttrHelper = WPDLLInt1.AttrHelper;
      IWPParInterface CurrPar = Memo.CurrPar;

   // set all margins, do not change page size
   Memo.PageSize.SetPageWH(-1,-1,360,360,360,360);

   // We want to use 2 different character styles
   AttrHelper.Clear();
   AttrHelper.SetFontface("Verdana");
   AttrHelper.SetFontSize(11);
   AttrHelper.IncludeStyles(1); // bold
   int headerchars = AttrHelper.CharAttrIndex;
```

```
   AttrHelper.Clear();
AttrHelper.SetFontface("Verdana");
AttrHelper.SetFontSize(8);
int bodychars = AttrHelper.CharAttrIndex;

// and add a one row table
TextCursor.AddTable("WPA",3,1,true,0,false,false);

// Now add one row after the other
string n = "",c ="",h ="";
for(int i=0; i<1000;i++)
  {
    if (!WPDLLInt1.Memo.wpaGetCaption(i,ref n, ref c, ref h)) break;
    TextCursor.CPMoveNextRow(true); // down and create
    TextCursor.CPTableColNr = 0;
    CurrPar.SetText(n, bodychars);
    TextCursor.CPTableColNr = 1;
    CurrPar.SetText(c, bodychars);
    TextCursor.CPTableColNr = 2;
    CurrPar.SetText(h, bodychars);
}

// add text to the header row
TextCursor.CPTableRowNr = 0;
TextCursor.CPTableColNr = 0;
CurrPar.SetText("Name", headerchars);
CurrPar.Alignment = 1;
CurrPar.ParShading = 30;
TextCursor.CPTableColNr = 1;
CurrPar.SetText("Caption", headerchars);
CurrPar.Alignment = 1;
CurrPar.ParShading = 30;
TextCursor.CPTableColNr = 2;
CurrPar.SetText("Hint", headerchars);
CurrPar.Alignment = 1;
CurrPar.ParShading = 30;

// and reformat
Memo.Reformat();
```

This is the created table:



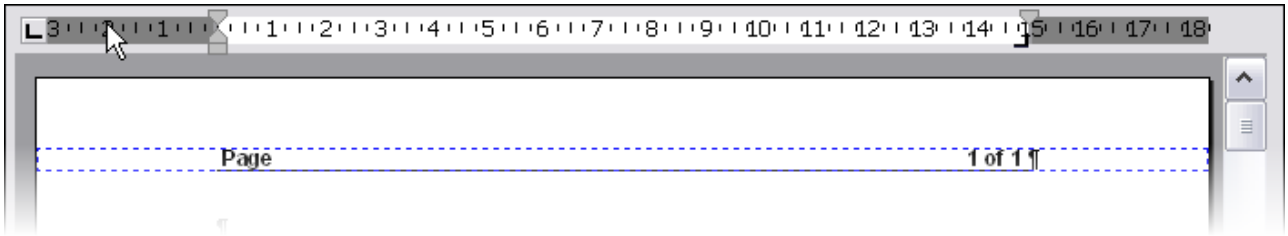
Name	Caption	Hint
ZoomOut	Zoom Out	Zoom Out
ZoomIn	Zoom In	Zoom In
BBottom	Bottom Borders	Bottom Borders
BInner	Inner Borders	Inner Borders
BLeft	Left Borders	Left Borders
BAllOff	Switch Borders Off	Switch Borders Off
BAllOn	Switch Borders On	Switch Borders On
BOuter	Outer Borders	Outer Borders
BRight	Right Borders	Right Borders
BTop	Top Borders	Top Borders
Bullets		Bullets

5.6 Create a page header with page numbers

This C# code generates a new header, a right tabstop and page numbering:

```
IWPMemo memo = WPDLLInt1.Memo;
IWPDataBlock header = memo.BlockAdd(DataBlockKind.wpIsHeader,
       DataBlockRange.wpraOnAllPages, "", 0);
if (header!=null)
    {
       header.Clear();
       header.WorkOnText = true;
       IWPTextCursor cursor = memo.TextCursor;
       memo.CurrAttr.IncludeStyles((int)(WPSTY.BOLD|WPSTY.UNDERLINE));
       cursor.InputText(" Page ");
       cursor.InputTabstop(false,
          memo.PageSize.PageWidth-
          memo.PageSize.LeftMargin-
          memo.PageSize.RightMargin,
          1 , 0);
       cursor.InputObject(TextObjTypes.wpobjTextObject,"PAGE","",0);
       cursor.InputText(" of ");
       cursor.InputObject(TextObjTypes.wpobjTextObject,"NUMPAGES","",0);
                 cursor.InputText(" ");
}
```

Result:



5.7 Example: Format C# Code

This C# example formats C-Sharp code in a very simple way.

It normalizes all paragraphs in the text. It also removes the character attributes from all characters and make the complete text use Courier New, 9pt. The code remove all spaces from the beginning of each paragraph and detect comments and nesting using { }. Depending on the nesting level tabs are inserted. We have applied this method to the code itself to format it.

```
IWPMemo Memo = wpdllInt1.Memo;
IWPTextCursor TextCursor = Memo.TextCursor;
IWPParInterface par = Memo.CurrPar;

TextCursor.CPPosition = 0;
int nesting = 0;
bool neststart;
while(par!=null)
```

```
{
   // Remove all paragraph and character attributes
   par.ParAClear(1);
   // and apply new attributes as paragraph attributes
   par.ParASet((int)WPAT.CharFont, par.ConvertFontnameToIndex("Courier New"));
   par.ParASet((int)WPAT.CharFontSize, 9 * 100); // ==9pt

   // Remove all spaces at the beginning
   while((par.GetChar(0)<=32)&&(par.GetChar(0)>0))
   par.DeleteChar(0,1);
   // This paragraph starts with // - make it green + italic since it is a comme
   if ((par.GetChar(0)==(int)'/')&&(par.GetChar(1)==(int)'/'))
   {
      par.ParASet((int)WPAT.CharColor, par.ConvertColorToIndex(0x00005E00));
      par.ParAAddBits((int)WPAT.CharStyleON, 2);
      par.ParAAddBits((int)WPAT.CharStyleMask, 2);
   }
   // else it is a closing }
   else if (par.GetChar(0)==(int)'}') nesting--;
   neststart = (par.GetChar(0)==(int)'{');
   // Insert tabs according to nesting
   for (int i = 0; i < nesting; i++)
   par.InsertText(0,"\t",-2);
   if (neststart) nesting++;
   // Move to next line
   if (!par.SelectNextPar(true)) break;
}
Memo.ReformatAll(true,true);

wpdllInt1.ReleaseInt(Memo);
wpdllInt1.ReleaseInt(TextCursor);
wpdllInt1.ReleaseInt(par);
```

5.8 HTML/E-MAIL loading and saving

5.8.1 Technical Information

TextDynamic can load and save HTML formatted files. It supports many HTML tags and also cascading style sheets (internally it uses a style class which mimics CSS style information). As a word processing engine it supports page layouts (header area, text area, footer area), a feature which is important for high quality printing of the documents, and can split table rows on several pages. Both features require a formatting routine which is different to the one used by a web browser. So usually websites will look differently when loaded into TextDynamic than they looked in a web browser. Frames are not supported.

HTML documents which are designed to be "printable" will usually look perfect in TextDynamic.

With Memo.SetBProp ist is possible to select a special HTML format mode! wpViewOptionsEx (Webpage Mode)

If you want to display a **split editor** with the HTML source (HTML code) and formatted text you can use TextCommandStr(5, ..) .

With an additional license it is now possible to load a **http access DLL** to automatically retrieve JPEG and PNG images referenced by HTML files.
This DLL is loaded by Memo.TextCommandStr(6,1, dll_key)

Notes:
a) Please see ActivateSyntaxHighlighter for XML and HTML syntax highlighting.

b) also see SetXMLSchema to work with XML tags

b) The format string "XML" writes simplified HTML code.
You can use "XML-useptag,-ignorefonts" to only write <p>, <b>, <i>, <u> and <a> tags. Paragraphs may use styles which are written as <p class="">.

5.8.2 Create HTML

You can create a HTML file by simply use the Memo.SaveToFile method. By default embedded images will be recreated using the naming
filename_000x.png or filename_000x.jpeg

This also happens when the save dialog is used. (Save or SaveAs action)

When the same file is saved another time in the same session the existing image file will be reused. The component will automatically compress bitmap (BMP) to PNG format if the color depth is <= 8 bit and otherwise JPEG format. The compressed image data will replace the embedded, non compressed image data.

In place of metafiles (EMF, WMF) a 200 dpi PNG image will be created. The PNG information will be only used to save the image, the embedded data will not be replaced.

If you use the format string HTML-imgpath:"" embedded images will not be exported. (This behaviour used to be the default setting)

If you want to save the images to a different directory use the format string
HTML-imgpath:"c:\attach\img"

Then the files
c:\attach\img_000x.png or c:\attach\img_000x.jpeg
will be created.

If you need to create HTML as string use the method SaveToString. Here the creation of image files must be explicitly activated using a format string like HTML-imgpath:"c:\attach\img".

Note: The double quotes are required for the -imgpath: format option!

5.8.3 Format Options

Using the format strings used by the IO methods the HTML reader and writer can be customized. The format strings are passed to the IO method as second or third parameter.

To select the HTML writer or writer use the format string

"HTML"

TextDynamic will usually detect HTML input correctly if the format string is empty or "AUTO" is used.

Options can be appended to the format string, the following options are useful for HTML:

For HTML writer:

-**imgpath:**"imagepath_and_filename"

This will tell the writer to create a copy of all images in the mentioned path. The filename part will be used to build the name for the image file.

-csspath:"http://www.wpcubed.com/docstyle.css"

Write a link to the give CSS file.

Examples:

"\" create name save using the path and name of the HTML file plus the number. (Does not work for SaveToStream and SaveToString)

"\testimages\" create files in sub directory *testimages* and use name _000x.

"" Do not create files. Only create IMG tags for linked images.

Default
For images which are not linked create image files using the name of the created HTML file + _000x.

-onlybody

Do not save the <html> and <body> tags to make it possible to use the output as text block in a website.

-writebasefont

Write a basefont tag using the information of the default attributes of the document.

-writespan

Write objects instead of <font>

For HTML reader:

-csspath:"c:\docstyle.css" Opens the local file "c:\docstyle.css" instead of the linked one.

-onlyinbodytag Ignore all text outisde of the tags <body>..</body>

-nospanobjects Don't create embedded SPAN objects

-ignorehtml do not use HTML formatting tags. this is useful in combination with -useBBCodes.

-useBBCodes Interpret the "bulletin board" tags [b], [i], [u], [s], [color], [size], [align], [center], [left], [right], [justify], [list], [url], [email]

-usecr Create a new paragraph at a carriage return character

-codepageXXXX Use codepage XXXX to load the text

For XML writer (see XML Mode):
-useptag Write <p> insetd of tags

-ignorefonts Do not write font information

5.8.4 Create and send Emails

Emails contain plain text, possible HTML text with images and further attachments. The creation and collection of all this data is not an easy task. If a document contains embedded images, first image files have to be created so the HTML part of the e-mail can link to them.

The TextDynamic **interface IWPMAPI** makes it easy to prepare the e-mail data. You are then free to either use the MapiSendMail function to send it (call Send) or you can read the properties and send the e-mail using a different tool. In this case you only have to interpret the list provided by AttachmentList.
When you execute Prepare the e-mail will be created, all attachments will be written. By default the e-mail will be created with the body as ANSI text and the document as HTML attachment. If you want the e-mail body to be HTML formatted text, first assign the string "<html>" to the property Body .

Important properties:

Body: The body as ANSI text, assign <html> to force HTML creation, assign "" to let the component create ANSI text. You can also assign some ANSI text will be then used as text body.

AddHTML: if true (default) the document will be appended as HTML attachment.

AttachmentList: the attachment files as list of filename=displayname pairs, delimited by comma. Prepare clears this list and adds the name of the HTML message and all images. If you want to add other files you can use AppendFile after preparing.

Important methods:

Prepare: Collect the data for an e-mail. This clears the attachment list.

Send: Send the e-mail which was prepared last. If no e-mail was prepared the document from editor 1 will be send.

Send2: Send the e-mail which was prepared last. If no e-mail was prepared the document from editor 2 will be send.

Clear: Resets the properties Subject, FromName, FromAddress, Recipients, CCList, BCCList, AttachmentList, Body.

Example (C#):

```
IWPMapi mapi;
mapi = wpdllInt1.MAPI;
if (mapi!=null)
{
    mapi.Recipients = "somebody@somewhere.com";
    mapi.AddCCRecipient("Julian", "julian@somewhere.com");
    mapi.AddCCRecipient("Claudia", "claudia@somewhere.com");
    mapi.Subject = "Test me";
    mapi.Body = "<html>";
    mapi.Prepare(1,0);
    MessageBox.Show(mapi.AttachmentList,
                    "MAPI created this attachments");
    mapi.Send();
}
```

5.8.5 Create MIME encoded e-mail data

The integrated MIME writer class can be used to create MIME formatted data:

If you use the method SaveToString with the format name "MIME" it will create multipart MIME e-mail data with the HTML code and all images embedded without a single temporary file being created!

A MIME message will be also written when the document is saved to a *.MSG file.

Example MIME Ouput Data:

```
From: Me@somewhere.com
To: somebody@somewhere.com
Subject: Test me
Date: Fri, 17 Nov 2006 09:00:29 +0100
CC: "Julian"<julian@somewhere.com>
MIME-Version: 1.0 (produced by Synapse)
X-mailer: WPTools/TextDynamic
Content-type: Multipart/mixed;
boundary="00010001_1855D7EE_MIME_Boundary"
Content-Description: Multipart message

--00010001_1855D7EE_MIME_Boundary
Content-type: text/html; charset=UTF-8
Content-Transfer-Encoding: Quoted-printable
Content-Disposition: inline
Content-Description: Message text

=EF=BB=BF<html>
<head></head><body>
```

```
<div><font face=3D"Arial">Some Text</font></div>
<div><font face=3D"Arial">and an image: <img
src=3D"CID000100000001" style=
=3D"width:0.23in;height:0.17in;"/></font></div>
</body></html>
--00010001_1855D7EE_MIME_Boundary

--00010001_1855D7EE_MIME_Boundary
Content-type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: Quoted-printable
Content-Disposition: inline
Content-Description: Message text

Some Text
and an image:  [image1.JPEG]
--00010001_1855D7EE_MIME_Boundary
Content-type: image/JPEG; name="image1.JPEG"
Content-Transfer-Encoding: Base64
Content-ID: CID000100000001
Content-Disposition: inline; FileName="image1.JPEG"
Content-Description: Included file: image1.JPEG
```


```
/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAMCAgMCAgMDAwMEAwMEBQgFBQQEBQoHB
wYIDAoM
DAsKCwsNDhIQDQ4RDgsLEBYQERMUFRUVDA8XGBYUGBIUFRT/2wBDAQMEBAUEBQkFB
QkUDQsN
FBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUF
BT/wAAR
CAAUABsDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/
8QAtRAA
AgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM
2JyggkK
FhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4e
XqDhIWG
h4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2
uHi4+Tl
5ufo6erx8vP09fb3
+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREA
AgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVY
nLRChYk
NOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3e
Hl6goOE
hYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2
Nna4uPk
5ebn6Onq8vP09fb3
+Pn6/9oADAMBAAIRAxEAPwDA+PfxTn8MeHxZ6PNfKz+TbhWj8uHczSZe
NwrEtjgjHQD1rzdfFHxJWGaC71S5uLeaTDrLPKmCCd2wlSF6dOM+lXviV8QYdcu9E
0+JZolt
riaYSMzFVwAoO0eucZ9q7y6/an1K00jzZ/ConuHuFURxXW0AfNycg8/
LnHvXk11ONnBHs4WW
Gkpe1lZnyvqukaff65c6f5U1ssRM6b0EbyyOcks2B+HHPNLa6DJYwJBJJtdOCJ13O
Pqe9dnq
et6Rr2sFJZWRkId2hOWjGTnkAE88dO1ePa4b2DWLtDFc4WQhcmfle3f0xXXGTa97c
8SpyKXu
6nvvijT4rnxddyuPmYlMKAoABJA4HufzrC1DU2fQ7i+MMXnWsqwouCVIc7SWBPJAH
```

```
HpRRWb2
FLdGPrF0ugW1q9jbwwzXhaFpgDujAzynOAfcg16NpXhOK7sUml1DUGkZnJPn4z8x9
qKKjodF
JK7P/9k=
--00010001_1855D7EE_MIME_Boundary--
```

The following format options can be used:

-noplain : Do not include the plain text
-nohtml : Do not include the text as HTML code. Embedded images will be still appended to the mail data.

In the plain text always placeholders for the images will be inserted, see the text "[image1.JPEG]" in the MIME example above.

Example:

```
html_mail = SaveToString(false, "MIME-noplain,")
```

The e-mail properties
Form, To, Subject, and CC will be set to the values assigned to the property MAPI!

The MIME writer uses functions of the free library Synapse at http://www.ararat.cz/synapse/

5.9 Use TextDynamic in C++ (VS2008)

In case You do not want to use .NET or the OCX Interface there is a third possibility to integrate TextDynamic in your application.

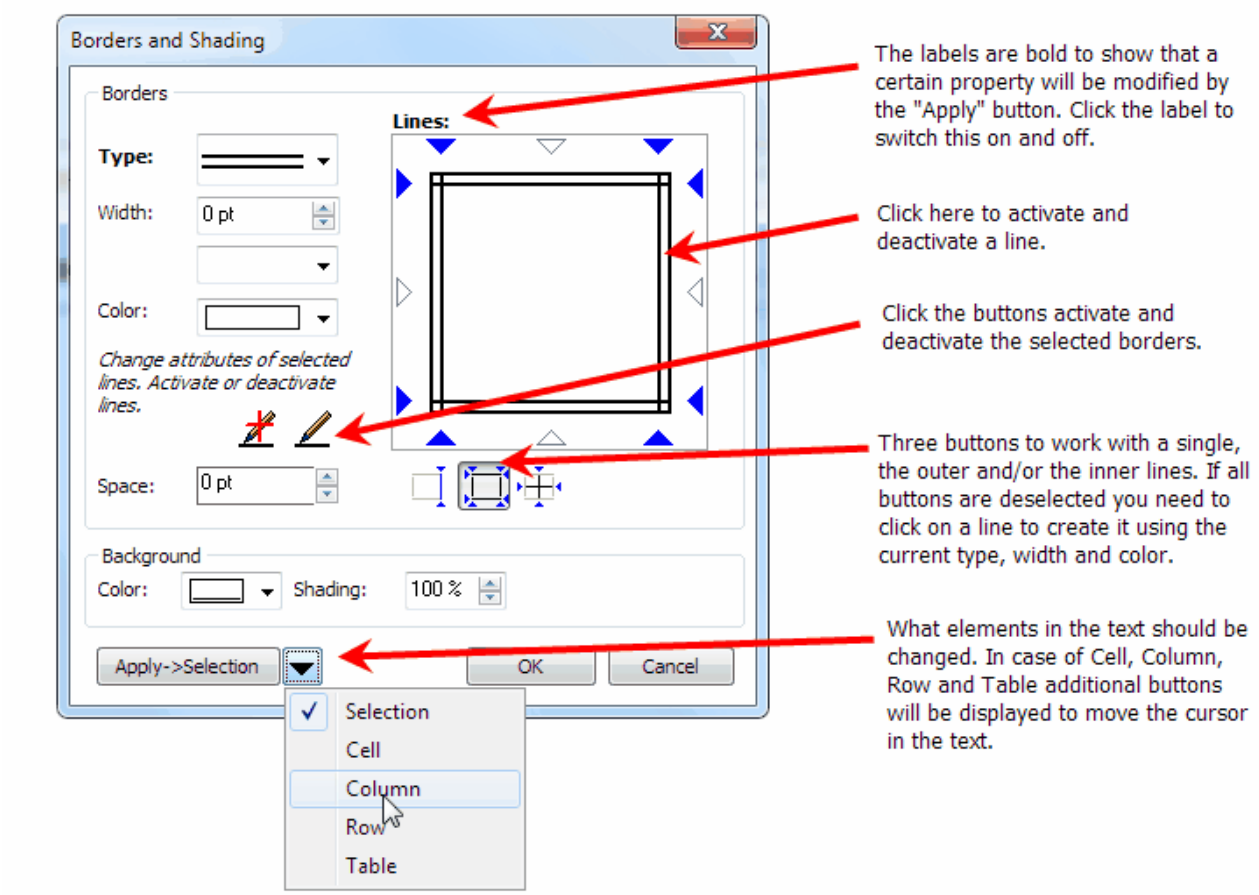
We call it the "Popup Editor Mode".

You can either create an editor dialog window, or place the editor inside one of your windows. In the latter case this will look like an embedded OCX, only that no COM is involved. You can still load and save the text.

Please read more here: TextDynamic as Popup Editor (method wptextCreateDialog)

5.10 The border dialog

This dialog will be displayed by Action wpaDiaParagraphBorder



5.11 Mailmerge (document variables)

Mail merge is the update and read out of data fields located in the document.

Please read here: Mailmerge and here Mailmerge Category

6 TextDynamic Release Notes

2.4.2014 Version 7
+ added unicode DLLs
+ added 64bit support (unicode DLL only)
+ improved PDF engine V4
+ new RTF Engine V7 with improvement5s to editing, rendering and HTML support

Note: Version Numbers 2-6 are skipped to keep up with the underlying RTF Engine version which is V7.
Upgrades to TextDynamic V7 are free, when you ordered on or after 1.10.2013.

30.10.2013 V1.96.7.1
* the bookmark show/hide did not work as expected
- some fixes to PDF engine
- small fix to css reader

15.7.2013 V1.96.2
+ IWPPrintParameter.AllPagePaperSource and FirstPagePaperSource values will now be saved to RTF.
They will only be loaded from RTF, when the format string "RTF-ReadPrintPar," was used. They are not loaded

in general, since the application must allow it to change the paper source.
(note: IWPPrintParameter is valid for the whole document and not for individual sections)
+ You can use IWPMemo.TextCommandStr(47) to retrieve a list of the tray names and IDs for the current printer.
+ The formatstring "RTF-ReadWPT4Fields," will make TD convert WPTools 4 fields into merge fields.
+ implements newer PDF engine
- fixes problem with the spellcheck directory scan for dictionaries

21.5.2013 V1.96
*** changed check for expiration of TextDynamic demo**
* modified convert TableToText method

11.3.2013 V1.95.1
- fix problem with wrong display of merge field start
* update of PDF engine improves font embedding

19.1.2013 V1.95
* RTF reader now handles UNC file links which use "\\" in the path
* the cursor was not painted if DoubleBuffered was set to true for the parent of the editor
+ Premium: Saves and loads \column
* stream RTFvariables were not loaded from WPT format. They are loaded now.

10.11.2012 V1.93
- fixes the problem that the print dialog did not change the selected printer
- Update to RTF reader to load landscape flag for sections better
- when page mirror was used, after a page break the text indentation was sometimes wrong
- hyphenation code was broken
- workaround for word files which have space characters in table definitions

19.10.2012 V1.92
- fixes problem with numbering of footnotes
- improvement to selection code (doubleclick)
- improvement to display of NL symbol (if active)
- improvement to tables with header+footer rows

20.6.2012 V1.91
- fixes problem with Demo DLL at startup
- includes some editor improvements.

19.3.2012 V1.90
NEW DEMO DLL - will reset the usage counter
* improved HTML export (i.e. write all parameters in "")
* improved HTML import (i.e. reading cell heights)
- when writing HTML background color is not set to white when shading is 0. 0 is treated as default value.
- image align center and right now works in HTML
- fix an endless loop when image was too large
- improvement of table border drawing
- improvement for right align in table cells
- RTF writer writes background color easier to understand by Word
* improved word wise cursor movement when fields are used
- dash symbols were not painted using the current font color
- some stability improvements
+ EditOptionsEx2: wpCellMultiSelect - allows multi selection in tables when CTRL is pressed
+ improved XML import/export (unit WPIOXML1.PAS)
- fix problem with TWPToolButton

- fix problem when painting insertpoints after tab stops. They were painted two times.
+ the text writer now understand the format option -softlinebreaks to create a \n at the end of
every line. In fact all soft line breaks will be handled like the #10 code.
- some smaller editor and stability problems fixed

18.10.2011 V1.78.1
+ added methods to delete, read and write data in fields (without having to use a callback)
+ enhanced MoveToField method to also select a field or move to its end.
+ It is possible to specify a field name inside the formatstring for the load and save functions: Load and Save Category

14.10.2011 V1.78
+ Sub paragraph: IWPParInterface.SetParType
- change in RTF reader to let section inherit the default layout, not the current page layout
- fix of problem with table borders when also PageMirror was used.
* updated border enhanced dialog
* updated border drawing code - now supports dotted lines with wider lines.
* modified method DeleteColumn
* modified WPT reading code to repair table width which were negative
+ improved image rendering code for transparent (PNG) images. They will be drawn transparently also when scaled and also in high resolution rendering mode.
+ new code to draw dotted lines which also supports wider lines
- premium license: fix problem when there were too columns
* MergeText now restores before Merge Cursor position and selection (except for cell selection)
* resizing a table column does not move the cursor to the nearby cell anymore
* different frame line when resizing columns and rows
+ InsertColumn now also works if wpAllowSplitOfCombinedCellsOnly was used in EditOptionsEx
+ improved paint routine now avoids clipping of characters which were overlapping their bounding box,
such as italic letters or "f".
The improvement is especially visible when selecting text or using character background colors
+ With premium license it is now possible to print a line between certain colums -
use par.ASet(WPAT_COLFLAGS, WPCOLUM_LINE);
+ paragraph styles can now contain border definition for paragraphs
* revised code to draw double borders - always draws two lines on screen even when zoomed
* improved saving of numbering attributes with styles
* style dialog can now apply number level even if style does not have numbering yet.
* fix problem with - - - - - at end of line
- fix problem with spell-as-you go after hyperlinks
- fix problem with page numbers in sections when tables were spanning pages
- fix problem with right aligned negative numbers in merge fields
* automatic text attribute was not inherited to tables inserted in fields
* images with mode "under text " can now be also clicked at positions where there is no text.

10.9.2011 V1.77
- + paragraph styles can now contain border definition for paragraphs
* revised code to draw double borders - always draws two lines on screen even when zoomed
* revised wpNoEditOutsideTable - was not checked for keyboard input
* fix problem with - - - - - at end of line
- fix problem with spell-as-you go after hyperlinks
- fix problem with page numbers in sections when tables were spanning pages
* automatic text attribute was not inherited to tables inserted in fields

20.7.2011 V1.75.4
* Memo.SetBProp(0, 23,1)can be used to force colored table borders to be printed black (workaround printer bug)
- updated PDF engine to sort named destination in lexical order

7.7.2011 V1.75.3
* outer border button (action) now also works for paragraphs
- IWPPageSize.GetProp was not working

* change to avoid flickering when doing auto scroll
- modified DBCS support for RTF reader

15.6.2011 V1.75.2
+ SetBProp(0,22) allows it to switch on a blinking caret
- font selector in small toolbar now uses smaller text height
- update to event OnTextObjectGetText to make it possible to update display of page numbers
- improved table handling in editor
- fixed problem in "WPT" reader to update current outlines with the loaded outlines
- update to PDF export
- resizing of images in the text improved (did not work if the image was made larger than the page)
* updated actions to apply inner and outer borders to selected cells
- update to symbol dialog, tab and table dialog
- fix problem with internal exception "too many EnableProtection".
* tripple click in margin selects paragraph
+ double click in margin selects current cell
+ tripple click in margin selects current row
- wpsec_ResetPageNumber flag is now saved in WPTOOLS format
* much improved border dialog

23.4.2011 V1.75
- fix selection problem when several images were linked to same paragraph
- when moving images the Z order will not be reset
- HTML Writer: A style with name "DIV" will be added to the style sheet to save the default font
- HTML Writer: BaseFont tag will now be written with font size (requires -writebasefont option)
- improved display of character background color for fields and other special code
* change in format routine to fix problem when a nested table cell caused a page break.
* several fixes and updates in editor
* PREMIUM: Improved editing of text boxes

25.9.2010 V1.72
+ new ViewOptionEx mode 13 to draw text starting with www, http and https as hyperlink. The hyperlink event can be used to capture the URL parameter on click.
* improved PDF export (requires PDF license): updated font embedding code
- improved RTF reader (problem with ANSI characters > #127)
- some improvement to table rendering code (bottom border of vertically merged cells)
+ HTML: new support for entity

24.7.2010 V1.71
* the print preview will now be displayed over main windows
* several improvements of editor
* improvement to RTF writer to when writing table cells
* improved right aligned text
* fixed problem with line heights of lines which are empty except for new line character
- fix for problem when pressing Accent + Backspace

17.6.2010 V1.69
+ new border dialog and improved handling and rendering of table and paragraph borders
+ improved text display
+ pasting text from FireFox works better
+ new ViewOptionsEx: wpShowCurrentCellAsSelected, wpShowCurrentRowAsSelected, wpShowCurrentTableAsSelected
+ it is now possible to load base64 embedded JPEGs from HTML
+ it is now possible to change width of tables which exceed right margin
+ Memo.UpdateDialog can now be used to hide the print setup buttons from the preview dialog

5.5.2010 V1.68.1
+ New command to convert tables into text
- fix in XML writer and RTF reader
- underlines are not drawn for spaces at line end anymore

5.4.2010 V1.67.8
* some improvements to RTF, HTML and XML reading and writing

28.3.2010 V1.67.5.1
+ Horizontal lines can now be colored. (InputObject)
+ TextCommandStr(38) can now be used to predefined the list of fonts displayed in the toolbars font drop down.
+ Command 26: Modify MergeText Operation
- some bug fixes
+ ViewOptionsEx: `wpInvertActiveRow and wpInvertActiveRowExceptForProtected`

8.3.2010 V1.67
+ Memo.SetBProp: ViewOptionsEx: wpDontExtendSelectionToRightMargin
+ Memo.SetIProp(11,-1) makes TextCursor.FindText search backwards
* improved text rendering. Fixes "moving character" problem.
- When several RTFData objects were use (Memo.RTFDataAdd) the premium addon was not working completely anymore.

3.3.2010 V1.66.4
- Ctrl+Left/Right now acts better with fields
* checked stability when used with MS Access
* improvement of optional http image fetch mode
* improvement of bullet and number support
+ wpa Action DiaInsSymbol can now start a non modal dialog with parameter "nonmodal"
+ new Memo.TextCommmandStr to read current paragraph, line and word

5.2.2010 V1.66.1
+ new wpa action GraphicRotateLeft and GraphicRotateRight to be used in graphic context menu
+ improvement to HTML writer

22. January 2010 V1.65
- HTML writer will write page info only if format string "-PageInfo" has been specified
- HTML writer will write &nspc; in all empty paragraphs
- Kerning mode is active by default (avoid effect of slightly moving characters when selecting text)
- GetObjAtXY function has been checked to work in Normal mode

22. December 2009 V1.64.5
- fix problem with keep in table rows
- update to .NET wrapper to support "Button" property in MouseUp/Down event
- wpa Action "CreateTable" will open the grid window at mouse position. Alternativley X,Y can be passed as string parameter

16. December 2009 V1.64
- fix problem with numbering
- XML tag mode can now also handle footnotes
- When a selection was replaced the attributes of the selection was not used for the new text. This problem has been fixed.
- InputObject can now be used for custom objects (ask WPCubed to provide necessary objects)

10. December 2009 V1.63
+ new commands in IWPParInterface.ParCommand to

find character attributes in paragraphs
to change certain attributes quickly,
to calculate the screen coordinate of a certain position within one paragraph.
- fix problem with paragraph numbering

5. November 2009 V1.62.3
+ OnFieldEnter/Leave can now also be used in regular editing mode. (Has to be activated)
+ possibility to use multiple watermarks. See topic Background Image / Watermark and Memo. Command 25: Select Watermark Image
- fix problem with encoding certain characters in RTF

3. November 2009 V1.62.1
- bugfixes to open property dialog on correct monitor on multi monitor environments
+ Command ID 33 - read and write EventButton properties
- activated the experimental RTL Support. Use Memo.SetBProp(8, 26, 1)

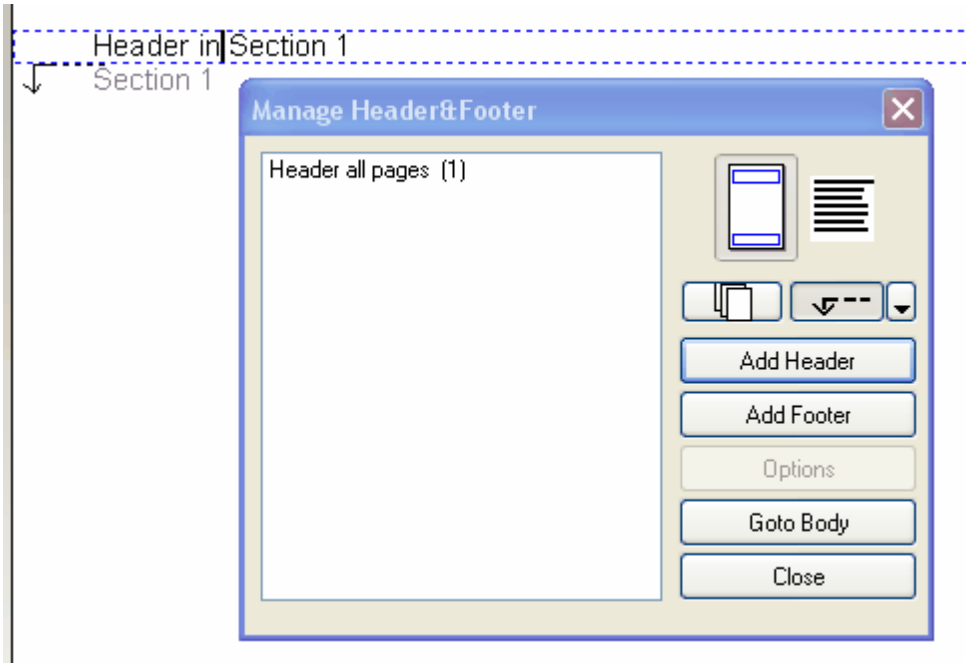
27. October 2009 V1.62
*** improved manual, see chapters Welcome, WPDllInt, IWPMemo, Getting Started and API Reference**
+ added topic: Text property reference
- fix operation of property Memo.Hidden
+ added Memo.SetBProp Group 17: EditOptionsEx2
+ now in vertical split mode, the thumbnails may be to the right of the editor. See SetEditorMode
+ option to modify the "address area". You can now also display a frame around the page.
+ option to set the minimum and maximum page count. (see Memo.Textommand 24: Draw lines and background, Page Size, Force Pagecount)
+ many additions to the possibilities to TextDynamic in "popup mode".
You can send all kind of actions to the editor and execute many commands.
+ faster monochrome dithered TIFF creation (premium or server)
- improvement of RTF reader
- improvement of PDF creator
* dialogs checked and controls aligned.
+ IWPParInterface (Memo.CurrPar) can now work with selected text, too:
Use SetProp(1,1) to work with selected text.
* improvement of in HTML writer

16. September 2009 V1.61
+ GetPageAsBitmap can now create a dithered monochrome TIFF file. (option)
please also see the new source code example which shows how to create a multi-page TIFF files
+ fix position problem for page image in footer

30. July 2009 V1.60.8.1
- change in PDF engine
- fix for hyperlink hover effect for links inside of table cells

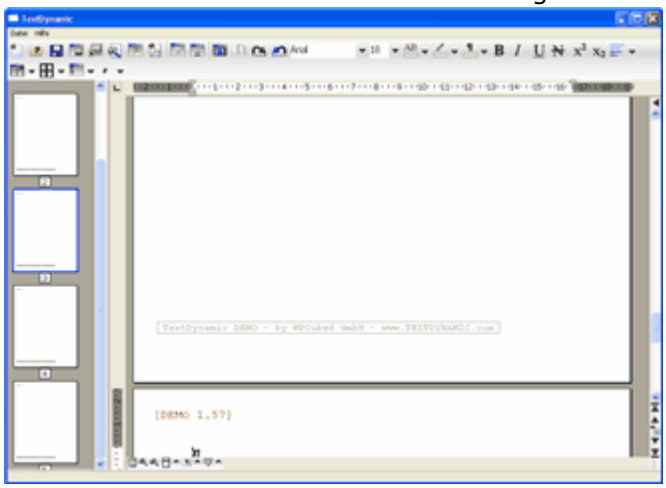
27. July 2009 V1.60.8
* when using "Delete All" in the tabstop dialog, all tabs will be cleared
added to manual: Tabstop Category
- fixed problem when deleting text in a paragraph. The alignment was cleared unexpectedly.
* improvement in PDF engine
+ we added the possibility to send commands to popup editor (i.e. in VS2008)
"With only a few lines of cods You can create a fully featured editor window in a C++ Application.
COM is not required, just link in the DLL and call one function"

20. July 2009 V1.60.5
+ improved manage header+footer dialog.
We added the possibility to create a new section and add header&footer to the current section.

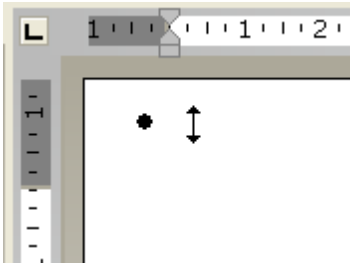


+ new KeepN support to keep paragraphs and tables together on a page. See Memo.SetBProp.
+ new FormatOptionEx2 wpfHideParagraphWithHiddenText to hide paragraphs which only contain hidden text.
* bugfix in PDF Generator
* RTF writer will now create different code for embedded SPAN objects to avoid problems when loading RTF in Word

10. July 2009 V1.60
+ **the DLL exports a new function "wptextCreateDialog"** which makes it easy to create an editor with a few lines of code.
Without having to deal with window classes, OCX or .NET!
+ new, automatic thumbnail modes
+ the two embedded editors can now be aligned horizontally. (See SetEditorMode)



* improved spellcheck engine now shows more suggestions.
* improved column handling (Premium Edition) with column balancing
* improved HTML loading and saving
+ support scrolling using the middle mouse button



* improved PDF export
* modified demo program
+ added method to create tabstops in the selected text

29. June 2009 V1.56
+ it is now possible to display a **gradient shading** effect on the "desktop" - see TextCommand
+ The reader now understand the format option "-**nonumstyles**" to load numbering not as style, but as text. This way the numbering can be represented as in the original (Word-) file. This is very useful when exporting RTF to PDF without any editing.
+ new ViewOptionsEx flags

26. May 2009 V1.55.3
* additions to manual - (VB6 notes)
- fix bug in spell check interface. This fix is important since the bug caused misbehavior especially in MDI applications
* setting of "SpellAsYouGo" will be set in new windows
* each window uses the same dictionaries but ca have its individual selected language
+ Memo.Reformat will also repaint the editor

22. May 2009 V1.55.3
+ new TextCommand to reformat the text and optional initialize all paragraphs in the next idle phase.
+ retrieve number of items on Undo Stack using TextCommand(19)
- fix in editor which caused an AV when part of a table was deleted

9. May 2009 V1.55
+ new in "Premium": SetXMLSchema to implement XML editor
* improved .NET wrapper. Now the data is preserved when the window is destroyed during a drag & dock operation
* improvement to HTML writer. Instead of the <font> the tag is written.

26. April 2009 V1.54.1
+ IWPSpell.SetProperty - is=1 can be used to hide the options tab in the options dialog
- fix problem with double spaces when Shift+Space key was pressed.
- improved spellcheck

25. March 2009 V1.54
*** improved spellcheck** (skip hidden text, improved spell compiler, fixed bugs, better handling in spell dialog)
*** improved display of text in editor with better character spacing.** (Using SetWYSIWYGMode You can disable the new mode)
* new formatting mode - use SetWYSIWYGMode(4) to activate it, 0 to deactivate.

13. March 2009 V1.53.3
* internal spellchecker has been checked. There is also a recompiled spell compiler.
* several improvements for the "ShowFields" mode
+ With UpdateDialog the buttons in the preview can be hidden
- some small problems have been fixed.

* scrolling with mouse wheel is now faster
+ SetBProp now has the possibility to clear all flags in a group

4. March 2009 V1.53.1

Much Improved merge field support. It is now possible to double click inside a filed to select the field and the start and end markers.
The Memo.ShowFields mode was also much enhanced. The cursor will not go inside the fields anymore and drag&drop inside the fields is avoided.

New API
IWPPdfCreator.Print, Export
IWPPdfCreator.PreparePDFAttachment

17. February 2009 V1.53

includes this Enhacements to the API:
IWPParInterface.DuplicateEx - i.e. copy tables rows.
IWPParInterface.TabCount - read tab positions
IWPParInterface.TabGet
IWPParInterface.TabMove
IWPParInterface.TabFind
IWPParInterface.TabGetNext
IWPParInterface.SelectFirstPar
IWPParInterface.SelectLastPar
IWPParInterface.SelectFirstParGlobal
IWPParInterface.SwapWithNextPar
IWPParInterface.SwapWithPrevPar
IWPParInterface.StartCPOffset
IWPParInterface.WidthTwips
IWPParInterface.HeightTwips
IWPParInterface.GetCharFontAt
IWPParInterface.GetCharSizeAt
IWPParInterface.GetCharStyleAt
IWPParInterface.GetCharColorAt
IWPParInterface.GetCharBGColorAt

IWPMemo:
GetPageAsBitmap - create PNG, BMP and, optionally, TIFF files from PDF pages
SaveStyleSheet
LoadStyleSheet
PrepareAttachmentList - for e-mail sending: makes it easier to create HTML output which a list of images files which also have to be attached.
GetAttachment
SaveNumberStyles
LoadNumberStyles
SetInitialPath
GetInitialPath

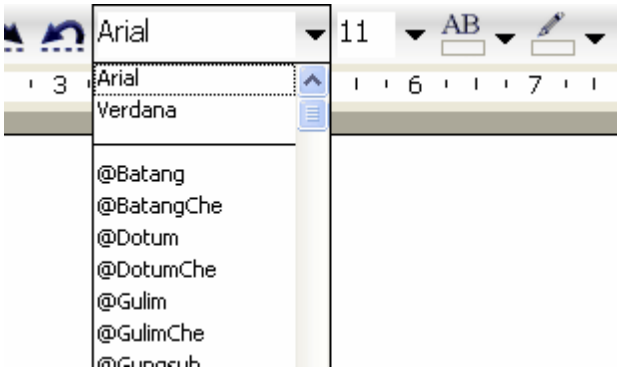
20. January 2009 V1.51.7
+ OnButtonClick is now called for all buttons, not only custom buttons.
+ use SetIProp(9, color) to set the default text color
- fixed probelm with TextCursor.InputPicture when the IPicture interface was passed
- some improvements in editor
- fix potential "method not found" problem on Windows 98 which was caused by the new InsertSymbol dialog

12. December 2008 V1.51.6
- fix problem which caused style dialog not to be displayed
+ add TextCommandStr to load and save styles sheets

9. December 2008 V1.51.5

* improved display of embedded Images
+ new SetIProp ID to set minimum caret width
+ new SetBRop to enable the word converter DLLs which may be installed on the system (Group 0)
* improvements to (optional) PDF engine
+ better column handling (Premium)

25. November 2008 V1.51
* set modified flag for change of font name, size and color
* the font selector in the toolbar now lists the 10 mostly used fonts in a document first. Cursor up/down now works. Return will select a font.



- fix of a problem which occurred when saving and loading certain texts
- when saving *.EML now MIME is written
* **embedded images will be drawn with anti aliasing** when scaled down (far better quality for embedded images)
- various fixes in editor engine

26. September 2008 V1.50
- better handling of movable images
- fix problem in InsertField
- some other stability issuse

19. September 2008 V1.49.5
- inserting an image object create the correct image object (fixes problem with linked images)
- premium: text objects are now movable
- improvements to LabelDef interface (property UsedUnits now works)
- improvement to MIME interface - created MHT file now compatible to IE

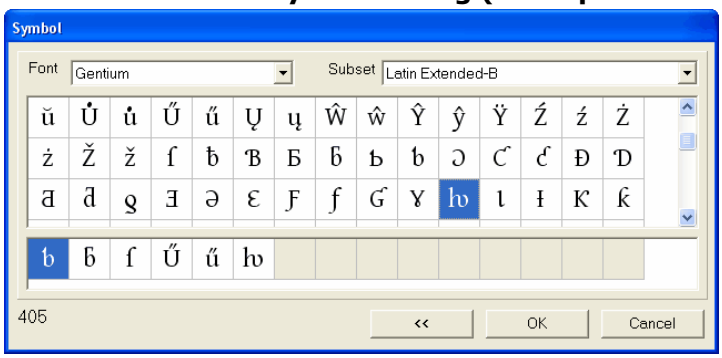
23. August 2008 V1.49.1
* bitmap export now allows different X and Y resolutions
* TIFF files are now compatible to FAX Viewer
* when exporting monochrome images colored text will be rendered black, shading will be ignored.

21. August 2008 V1.49
+ BMP export
+ PNG export
+ TIFF export.
- fixes problem in HTML reader
- EMF export did not use correct page size when normal layout mode was used.

10. June 2008 V1.46.5
+ EMF data is now saved to RTF as "emfblip" data.

2. June 2008 V1.46

+ extended insert symbol dialog (use wpa action `DiaINSSymbol`):



*** improved editor**

27. March 2008 V1.45
*** revised manual (see mail merge, reporting, TextCommandStr and Introduction)**
+ added token to template conversion
+ added HTML and field token syntax highlighting
+ wpa action tabletotext can now be used.
if parameter is empty string: separate by tabs, multi lines
"PAR" or Char(13): Create paragraphs
"TAB" or Char(9): Separate by tabs, single line
other: separate with string
- CurrSelAttr was not always updated. This has been fixed.

Please read the updated FAQ and the updated Reporting Manual.

27. February 2008 V1.43
+ added support for Visual Studio 2008
please use the assembly WPTDynInt3.dll
+ it is now possible to display a style drop down list in the toolbar:

```
<dropdownlist width="150" Image="-1" text="" wpa="ParStyleSelection"/>
```

20. January 2008 V1.42.1
- HTML reader created extra paragraph before table sometimes
- right click on misspelled word opened 2 popups
+ memo.DeleteParWithCondition can now optionally just hide the paragraphs
+ new TextCommand(15) to hide all text (also multiple paragraphs) inside fields when also property Memo.ShowFields = true

14. January 2008 V1.42
* If you use the event OnMouseDownWord to show a custom popup dialog please use Memo.SetBProp(0, 19, -1) to deactivate the default popup dialog.
- IWPFieldContents.ContinueOptions was not working.
+ new actions: wpaPasteText and wpaPasteSimple
* it is now possible to add actions such as "bold" to the popup menu in the PCC file:
Example:

```
<menues>
    <popup name="standard">
      <menu wpa="Bold"/>
```

+ for popup menus the parameter ifselected=yes/no has been added.

17. December 2007 V1.41
+ new TextCommand(13) to combine ALL adjacent tables
+ new autodetection of UTF8 HTML text (reading first 3 characters)
* improved cursor up/down movement routines
- fixed PDF export
- bug fix in HTML reader
+ updated BUTTONS.PCC - now showing a button for image insertion
+ new option to hide all table borders, see Memo.SetBProp

30. November 2007 V1.40
+ added chapter in this manual Configure the Editor
* the chapter SetBProp was updated.
* several improvements to editor
* fixed saving of spacing inside table cells
+ new PrintOption - print all colors in black (SetBProp)

14. November V 1.39
- fix in InputRowStart - Border was not applied
- sever fixes in RTF handling

25. September 2007 - V 1.38
+ The methods IWPTextCursor.GetParName and SetParName can be used work with current table name.
+ ParStrCommand can be used to update the name of the current table
- updated text processing
* when pasting ANSI text the current paragraph attributes are applied to all new paragraphs
+ new experimental AsWebPage display mode. Activated with Memo.SetBProp(7,0,1). Can be combined with HTTP loading option.

19. July 2007 - V 1.37
+ new method DrawToBitmap (.NET only) to take a screenshot of the editor.
- LoadFromString always inserted text (Insert parameter was ignored)

29. June 2007 - V1.36.2
* re-done and extended glyph set - see Provided Glyphs. (the images are included in file Buttons.pcc)
- bug fix in method GetPageAsMetafile
- bug fix for OnPaintWatermark event

+ support for small capital character style (use Action <-Check Image="151" wpa="SmallCaps"/>)
- Memo.EnumSelParagraphs did not work when control pars were selected, too
* IWPPageSize.SetProp is automatically called when a section property is changed to select the modified value.
+ new "floppy" save button image in buttons.pcc
+ TextCursor.InputSection can now select the current section to modify the current page attributes
+ Additions to TextCursor.CheckState
- Memo.EnumParagraphs did not use the EventParam parameter.

5. June 2007 - V1.35.6
+ RTFDataAppendTo can now create sections
+ when loading HTML you can specify the format string "-utf8" if the HTML code contains UTF8 characters
- better handling of code pages when saving HTML and RTF
+ IWPPageSize.SetProp can be used to select properties for a new section
* HTML reader is more forgiving when loading non well formed X-HTML
- some fixes for editing problems (undo)
* ReleaseInt now makes sure not to delete interfaces which are still needed

23. April 2007 - V1.35
* the event OnPaintWatermark was updated and it now receives the page number in "Mode"
+ new ID in ParCommand to check if it a paragraph is empty is empty

+ new ID in Memo.TextCommandStr to mark certain paragraphs to be exported as PDF outlines (bookmarks)
- CurrAttr.GeFontface did not work
- fix in Memo.GetXY . the current X,Y coordinate was not correct
* improved PDF engine

17. March 2007 - V1.34
+ improved .NET assembly. IWPMemo is now implemented as C# class and most interface references are automatically managed. (See ReleaseInt)
+ new internal HTML syntax highlighter - activate it with Memo.TextCommand(12,1,0)
+ new source view (ie "HTML" source) - use Memo.TextCommandStr(5, 1, "HTML")

28. February 2007 - V1.31
+ New .NET method ReleaseInt
* the hyperlink detection now moves trailing dots after the link
+ The .NET method SetDLLName will now pre-load the engine. This improves the speed when a lot editors are created dynamically.
+ New API: IWPTextCursor.TableSplit
IWPTextCursor.TableSort sorts rows in a table
IWPTextCursor.ASetCellProp changes attributes of a range of cells
IWPTextCursor.ASetCellStyle
IWPTextCursor.MergeCellHorz
IWPTextCursor.MergeCellVert
IWPTextCursor.ClearAllHeaders
IWPTextCursor.ClearAllFooters

9 February 2007 - V1.30.1
- SelectPrinter now works
- Custom Spellcheck now works without license
- Improvements to drag&drop and image handling

30 January 2007 - V1.30
* much improved manual - it now incorporates the developers API Reference
+ **the .NET assemblies are now all "strongly named".**
This makes a recompilation of the projects necessary.
+ the included DLLs and the OCX are now signed by WPCubed GmbH (Authenticode Technology)
+ you can create custom actions in the PCC file.
- better drag&drop handling
- improved PDF export (the CID font feature works better)
- possibility to embed binary data in the PDF file
+ new wpa action: DiaFontSelect - shows a dialog to change font name and style
+ TextAttr is now also published by interface IWPMemo
+ new method MoveToField
+ new method InputEmbeddedData - you can insert objects for PDF attachments (requires the optional PDF creation license)
+ new low level methods to move current paragraph (IWPParInterface.Select..)
+ new convert methods to calculate index values from text, font name and color values to be used with ParASet
+ new low level methods to loop all paragraphs in a document. (ActiveText.SelectFirstPar, SelectNextPar)

15 December 2006 - V1.29.3
- problem when using scroll wheel and pressing mouse button has been fixed.
+ automatic hyperlink detection now also works with email addresses

4 December 2006 - V1.29.2
- the internal paste from clipboard function was called twice for the CTRL+V key, this has been fixed.
* when saving HTML the embedded images will now be saved using a relative path

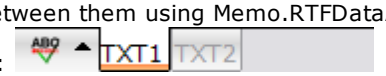
24 November 2006 - V1.29.1
+ improved method to extract metafiles: Memo.SavePageAsMetafile and Memo.GetPageAsMetafile
+ new property PageSizeList to install a list of page size definitions (see demo "ExtractMetafiles")

17 November 2006 - V1.29
* the internal interfaces (Memo, TextCursor) have been enhanced to be "dual". This improves the compatibility

with development systems which use the OCX version of TextDynamic.
+ when saving HTML automatically embedded images will be written as files. See chapter "HTML loading and saving"
+ the spellcheck dialogs can be now localized suing the XML code in the PCC file
+ the **interface MAPI** can be used to create e-mails (HTML incl. attachments) and send them
+ the **integrated MIME** encode can be used to create e-mail data (use format string MIME)
- The PCC file has been updated. The german language strings are now complete. The save button on the toolbar uses the "DiaSaveAs" instead of "DiaSave" action.
* the demo "Simulated MDI" is now also available as VB.NET project

7 November 2006 - V1.28
+ **exciting new label feature** (creation, preview and print)
+ Memo.RTFDataAppendTo to create a large text from multiple copies of the current
+ Memo.SaveToVar - save the text to a variant (.NET: "object") - for better performance
+ Memo.LoadFromVar - load data from a variant
- fix - buttons for actions such as "bold" were not updated

31 October 2006 - V1.27.7
+ new Simulated MDI demo. Instead of using a MDI (multiple document interface) you can store multiple documents into one editor and switch between them using Memo.RTFDataAdd, -Select and -Delete.

+ Also an tabset (panelH2) can be used: 

- API wpaSetFlags was not working, it has been fixed and a new manual entry has been added.
* improved CMH file (API reference)

28 October 2006 - V1.27.6
- event OnUpdateGUI was triggered too often due to an unexpected windows message
- Changing the column size of tables was switched off - it is now activated again
* please see new chapter "Create MDI Application (VB.NET)"

26 October 2006 - V1.27.5
* new shaded look and other designs for toolbars:  - reset with Command(9502,3,0)

+ Reporting interface is now complete
- backspace and cursor keys now work when pressed, not when released
* "ref bool" in TextCursor.FindText was never changed. Fixed.
* TextCursor.WordEnum now also operates correctly if the word was replaced in the event OnEnumParOrStyle

10 October 2006 - V1.26
+ **OCX** interface further improvements. It prints itself in a MS Access form!
+ new property SpellCtrl. The new interface IWPSpell allows it to load and select dictionaries for the integrated (optional) spellcheck engine.
+ new groups 13 and 14 for Memo.SetBProp. Now you can disable save and copy to clipboard operations. It is possible to allow copy&paste only within the application.
+ new properties: EventField and EventButton - for access in case your development system does not allow it to use the interface references passed to events.

15 September 2006 - V1.25
+ **OCX** interface completely redone for better operation in MS Access!
    + new property Readonly
    + new method SetLayoutMode
    * property Text now creates a VARIANT ARRAY OF BYTES - it will also accept the assignment of strings.
    - event OnInitControl works
    - event OnChangingText does work
    - property TextFormat now persistent
    + new editor for user define property

"**InitScriptXML**" - initialize the editor without a single line of code in VB6!
    * poperty **DLLName** now persistent
+ new method TextCursor.ScrollToCP - to show current line at top position
+ prepared new interface "IWPMapi" - this interface will allow in a later release (probably V1.30) allow the creation and the sending of mails. Currently this interface is not used!
+ advanced PDF engine: support for unicode (CIDFONTS) and PDF/A
+ popup menues can now be also specified in the PCC package file
- function TextCursor.FindText did not work reliably when boolean FromStart was not true
- security checks in interface case to avoid problems when commands were used while the editor was in loading state
- fix to avoid delay when typing first character

- fix to avoid ruler flickering when moving from record to record in MS Access

25 August 2006 - V1.22
+ **.**NET assembly now supports new API:
  public void SetBytes(int Editor, byte[] Text, string Format, bool Insert)
  public byte[] GetBytes(int Editor, bool OnlySelection, string Format)
  You can use it to load and save the contents into a bytes array instead of a unicode string.
+ Several improvements to editing and update of states in toolbar

15 August 2006 - V1.21
+ **PDF creation** is now always available (without license in unregistered mode).
  Please try it out using: `wpdllInt1.wpaProcess("DiaExportToPDF","");`
* improved internal exception handling
* improved word processing engine (mainly RTF reading part)

22 July 2006
* improved image support, load&save of PNG images
* some improvements to API
- fix problem in method LoadFromString - parameter "Insert" was ignored

6 June 2006
+ new: IWPMemo.GetNumberStyle to get interface to work with number styles
+ new interface IWPNumberStyle

19 May 2006
+ new: IWPMemo.GetMouseXY to locate current mouse position on page
+ new: IWPMemo.GetObjAtXY to locate an object at or near a certain position.
+ new: IWPMemo.GetPosXY locate the CPPosition at a certain x,y position
+ new: IWPMemo.GetXY get 15 value pairs (see CMH file!)
+ new: IWPTextObj.ShowHint - display hint at object
+ new: IWPTextObj.ObjType - read type of object

8 May 2006
+ easy creation of menu items (see C# demo)
- header/footer were not displayed
+ better handling of the Ctrl+Cursor keys
* better resizing of images
* larger dropdown arrows for toolbuttons

1. May 2006
+ new event OnUpdateGUI to make it easy to use own toolbar and menus
+ method wpaGetFlags to get the enabled, selected and hidden state of all "wpa" Actions as one array
+ facelift to ImagePack and TestApp
+ completed interfaces, fixed bugs
+ revised OCX (You will need to recreate it in your VB6 code)

15. March 2006
TextDynamic V0.95 - now also includes .NET support developed in C#. There are 2 DLLs, one for framework 1.1 and one for Framework 2.0. Also thanks to your comments and suggestions we were able to improve the programming interface and make it much easier to use. TextDynamic V0.95 has been tested with VB6, VS.NET 2005 and Delphi 2006.

The following functionality has been added:
+ PDF export
+ SpellCheck
+ Mailmerge
+ several interfaces to create text and tables under program control

14. February 2006
TextDynamic V0.90 - a powerful word processing DLL/OCX goes into beta test. At first for Visual Basic 6, later other development systems will be supported.

7 RTF2PDF Release Notes

23.9.2014 - V4.0.1.1
- fix problem with text boxes in RTF text
- improved transparency support of PDF engine

- improved Type3 creation of PDF generator
- several smaller enhancement to HTML loading

10.8.2014 - V4.0.1
*** update of our ASP demo at http://178.77.68.45/ (32 and 64 bit on Windows 2008 Server)**
+ added chapter "ASP .NET Troubleshooting"
+ Add Copyright info the the PDF properties (see SetProp)
+ Add extensions to the PDF XMP (see SetProp)
+ finetune Type3 embedding - only add certain fonts (see SetProp)

25.3.2014 - V4.0
+ added unicode DLLs
+ added 64bit support (unicode DLL only)
+ improved PDF engine V4 (wPDF 4)
+ new RTF Engine V7 with improvement5s to rendering and HTML support
+ new PDFCreator.Fontmode EmbedType3Fonts

+ IWPPrintParameter.AllPagePaperSource and FirstPagePaperSource values will now be saved to RTF.
They will only be loaded from RTF, when the format string "RTF-ReadPrintPar," was used. They are not loaded
in general, since the application must allow it to change the paper source.
(note: IWPPrintParameter is valid for the whole document and not for individual sections)
+ You can use IWPMemo.TextCommandStr(47) to retrieve a list of the tray names and IDs for the current printer.
+ The formatstring "RTF-ReadWPT4Fields," will make TD convert WPTools 4 fields into merge fields.
- fix problem with wrong display of merge field start
* RTF reader now handles UNC file links which use "\\" in the path
+ Saves and loads \column
* stream RTFvariables were not loaded from WPT format. They are loaded now.

Upgrades to RTF2PDF/ TextDynamic Server V4 are free, when you ordered on or after 1.10.2013.

10.11.2012 V3.93
- Update to RTF reader to load landscape flag for sections better
- when page mirror was used, after a page break the text indentation was sometimes wrong
- hyphenation code was broken
- workaround for word files which have space characters in table definitions
- fixes problem with numbering of footnotes
- improvement to selection code (doubleclick)
- improvement to display of NL symbol (if active)
- improvement to tables with header+footer rows

19.3.2012 V3.90
* improved HTML export (i.e. write all parameters in "")
* improved HTML import (i.e. reading cell heights)
- when writing HTML background color is not set to white when shading is 0. 0 is treated as default value.
- image align center and right now works in HTML
- fix an endless loop when image was too large
- improvement of table border drawing
- improvement for right align in table cells
- RTF writer writes background color easier to understand by Word
* improved word wise cursor movement when fields are used
- dash symbols were not painted using the current font color

- some stability improvements
+ improved XML import/export
- fix problem when painting insertpoints after tab stops. They were painted two times.
+ the text writer now understand the format option -softlinebreaks to create a \n at the end of
every line. In fact all soft line breaks will be handled like the #10 code.
- some smaller editor and stability problems fixed
+ It is possible to specify a field name inside the formatstring for the load and save functions: Load and Save Category

14.10.2011 V3.78
+ Sub paragraph: IWPParInterface.SetParType
- change in RTF reader to let section inherit the default layout, not the current page layout
- fix of problem with table borders when also PageMirror was used.
* updated border enhanced dialog
* updated border drawing code - now supports dotted lines with wider lines.
* modified WPT reading code to repair table width which were negative
+ improved image rendering code for transparent (PNG) images. They will be drawn transparently also when scaled and also in high resolution rendering mode.
+ new code to draw dotted lines which also supports wider lines
- fix problem when there were too columns
* MergeText now restores before Merge Cursor position and selection (except for cell selection)
* resizing a table column does not move the cursor to the nearby cell anymore
* different frame line when resizing columns and rows
+ InsertColumn now also works if wpAllowSplitOfCombinedCellsOnly was used in EditOptionsEx
+ improved paint routine now avoids clipping of characters which were overlapping their bounding box,
such as italic letters or "f".
+ paragraph styles can now contain border definition for paragraphs
* revised code to draw double borders - always draws two lines on screen even when zoomed
* improved saving of numbering attributes with styles
- fix problem with page numbers in sections when tables were spanning pages
- fix problem with right aligned negative numbers in merge fields
* automatic text attribute was not inherited to tables inserted in fields

20.7.2011 V3.75
* Memo.SetBProp(0, 23,1)can be used to force colored table borders to be printed black (workaround printer bug)
- updated PDF engine to sort named destination in lexical order
- IWPPageSize.GetProp was not working
- modified DBCS support for RTF reader
- improved table handling in editor
- fixed problem in "WPT" reader to update current outlines with the loaded outlines
- update to PDF export
- resizing of images in the text improved (did not work if the image was made larger than the page)
* updated actions to apply inner and outer borders to selected cells
- update to symbol dialog, tab and table dialog
- fix problem with internal exception "too many EnableProtection".
- wpsec_ResetPageNumber flag is now saved in WPTOOLS format
- HTML Writer: A style with name "DIV" will be added to the style sheet to save the default font
- HTML Writer: BaseFont tag will now be written with font size (requires -writebasefont option)
- improved display of character background color for fields and other special code
* change in format routine to fix problem when a nested table cell caused a page break.
* several fixes and updates in editor
* PREMIUM: Improved editing of text boxes

25.9.2010 V3.72
* improved PDF export : updated font embedding code

- improved RTF reader (problem with ANSI characters > #127)
- some improvement to table rendering code (bottom border of vertically merged cells)
+ HTML: new support for entity
* the print preview will now be displayed over main windows
* several improvements of editor
* improvement to RTF writer to when writing table cells
* improved right aligned text
* fixed problem with line heights of lines which are empty except for new line character

17.6.2010 V3.69
+ new border dialog and improved handling and rendering of table and paragraph borders
+ improved text display
+ it is now possible to load base64 embedded JPEGs from HTML
+ it is now possible to change width of tables which exceed right margin
+ Memo.UpdateDialog can now be used to hide the print setup buttons from the preview dialog

17. June 2010 V3.76
+ it is now possible to load base64 embedded JPEGs from HTML

3. March 2010 V3.74.3
* improvement of optional http image fetch mode
* improvement of bullet and number support
- HTML writer will write page info only if format string "-PageInfo" has been specified
- HTML writer will write &nspc; in all empty paragraphs
- fix in event handling

22. December 2009 V3.74
- fix problem with keep in table rows
* XML reader/writer handles foot notes
+ new commands in IWPParInterface.ParCommand to
  find character attributes in paragraphs
  to change certain attributes quickly
- fix problem with paragraph numbering
- fix problem with encoding certain characters in RTF

28. October 2009 V3.73
*** improved manual, see chapters WPDllInt, IWPEditor, and API Reference**
+ added topic: Text property reference
+ faster monochrome dithered TIFF creation (requires server license)
- improvement of RTF reader
- improvement of in HTML writer
- improvement of PDF creator
+ IWPParInterface (Memo.CurrPar) can now work with selected text, too:

31. July 2009 V3.72
+ Use SetSProp(7, filename) to specify a background image for all files
+ updated PDF engine
+ new KeepN support to keep paragraphs and tables together on a page. See Memo.SetBProp.
+ new FormatOptionEx2 wpfHideParagraphWithHiddenText to hide paragraphs which only contain hidden text.
* bugfix in PDF Generator
* RTF writer will now create different code for embedded SPAN objects to avoid problems when loading RTF in Word
+ The reader now understand the format option "-**nonumstyles**" to load numbering not as style, but as text. This way the numbering can be represented as in the original (Word-) file. This is very useful when exporting RTF to PDF without any editing.

+ new ViewOptionsEx flags (sew SetBProp)
* improved column handling (Premium Edition) with column balancing
* improved HTML loading and saving

23. August 2008 V3.69
+ BMP export
+ PNG export
+ TIFF export.
- fixes problem in HTML reader
- EMF export did not use correct page size when normal layout mode was used.

24. July 2008 V3.68
* updated wrapper for VS2008 (also see the trouble shooting note)
* improved PDF export engine
+ EMF data is now saved to RTF as emfblip data.#

18. April 2008 V3.66
+ add PDF tags to mark paragraphs and tables
- fix resource leak which occurred with some PNG Images were used.
- improve PDF export engine

28. March 2008 V3.65
+ new, optional reporting is available now
* improved PDF engine
* improved formatting performance
* revised manual (see mail merge, TextCommandStr and Introduction)
+ added syntax highlighting (HTML)
- CurrSelAttr was not always updated. This has been fixed.

14. January 2008 V3.60
+ new printing function - see Commands to print text (to printer, not PDF!)
+ HTML rendering mode

30. November 2007 V3.40
* several improvements to word processing engine
* better handling of Japanese unicode characters when exporting PDF
* fixed saving of spacing inside table cells

14. November V3.39
- fix in InputRowStart - Border was not applied
- sever fixes in RTF handling

20. July 2007 - V3.37
* improved support for keepN.
* some improvements to RTF engine
* better handling of auto sized tables - needs to be activated using Command(1029,1)

5. June 2007 - V3.35.6
+ RTFDataAppendTo can now create sections
+ when loading HTML you can specify the format string "-utf8" if the HTML code contains UTF8 characters
- better handling of code pages when saving HTML and RTF
+ IWPPageSize.SetProp can be used to select properties for a new section
* HTML reader is more forgiving when loading non well formed X-HTML
* ReleaseInt now makes sure not to delete interfaces which are still needed

19. April 2007 - V1.52
+ improved .NET assembly. IWPEditor is now implemented as C# class and most interface references are automatically managed
+ the event OnPaintWatermark can be used now. It receives the page number in "Mode".

+ new ID in ParCommand to check if it a paragraph is empty is empty
+ new ID in Memo.TextCommandStr to mark certain paragraphs to be exported as PDF outlines (bookmarks)

28. February 2007 - V1.51
New .NET method ReleaseInt - this releases an interfaces at once and avoids problem with garbage collection to access an interface although the object has been already disposed.
New API: IWPTextCursor.TableSplit
IWPTextCursor.TableSort sorts rows in a table
IWPTextCursor.ASetCellProp changes attributes of a range of cells
IWPTextCursor.ASetCellStyle
IWPTextCursor.MergeCellHorz
IWPTextCursor.MergeCellVert
IWPTextCursor.ClearAllHeaders
IWPTextCursor.ClearAllFooters

16. February 2007 - V3.50.5
+ new ASP.NET test server is online at http://www.rtf-net.com
+ "Premium" features now included (TextBox, footnotes)
- several improvements in API (InputHeader, SelectNextPar)
- InputHyperlink initialized "title" instead of "href"
- improvement to work with Windows 64 (metafile fix)
- solved threading issue

10. February 2007 - V3.50
+ first public release of RTF2PDF V3.5, akn "plus"
+ you can load license info from file
+ new setup procedure
* changed license keys - code looks like "wwww-xxxx-yyyy-zzzz"

30 January 2007 - V3.49 (beta)
* much improved manual - it now incorporates the developers API Reference
+ **the .NET assemblies are now all "strongly named".**
This makes a recompilation of the projects necessary.
+ the included DLLs and the OCX are now signed by WPCubed GmbH (Authenticode Technology)
- improved PDF export (the CID font feature works better)
- possibility to embed binary data in the PDF file
+ TextAttr is now also published by interface IWPMemo
+ new method MoveToField
+ new method InputEmbeddedData - you can insert objects for PDF attachments (requires the optional PDF creation license)
+ new low level methods to move current paragraph (IWPParInterface.Select..)
+ new convert methods to calculate index values from text, font name and color values to be used with ParASet
+ new low level methods to loop all paragraphs in a document. (ActiveText.SelectFirstPar, SelectNextPar)
+ improved method to extract metafiles: Memo.SavePageAsMetafile and Memo.GetPageAsMetafile
+ new property PageSizeList to install a list of page size definitions (see demo "ExtractMetafiles")
* the internal interfaces (Memo, TextCursor) have been enhanced to be "dual". This improves the compatibility with development systems which use the OCX version of TextDynamic.
+ when saving HTML automatically embedded images will be written as files. See chapter "HTML loading and saving"
+ the spellcheck dialogs can be now localized suing the XML code in the PCC file
+ the **interface MAPI** can be used to create e-mails (HTML incl. attachments) and send them
+ the **integrated MIME** encode can be used to create e-mail data (use format string MIME)
+ Memo.RTFDataAppendTo to create a large text from multiple copies of the current
+ Memo.SaveToVar - save the text to a variant (.NET: "object") - for better performance
+ Memo.LoadFromVar - load data from a variant

8 Info: WPViewPDF - View, merge and split PDF files

To view, print and manipulate PDF files right in your Application our

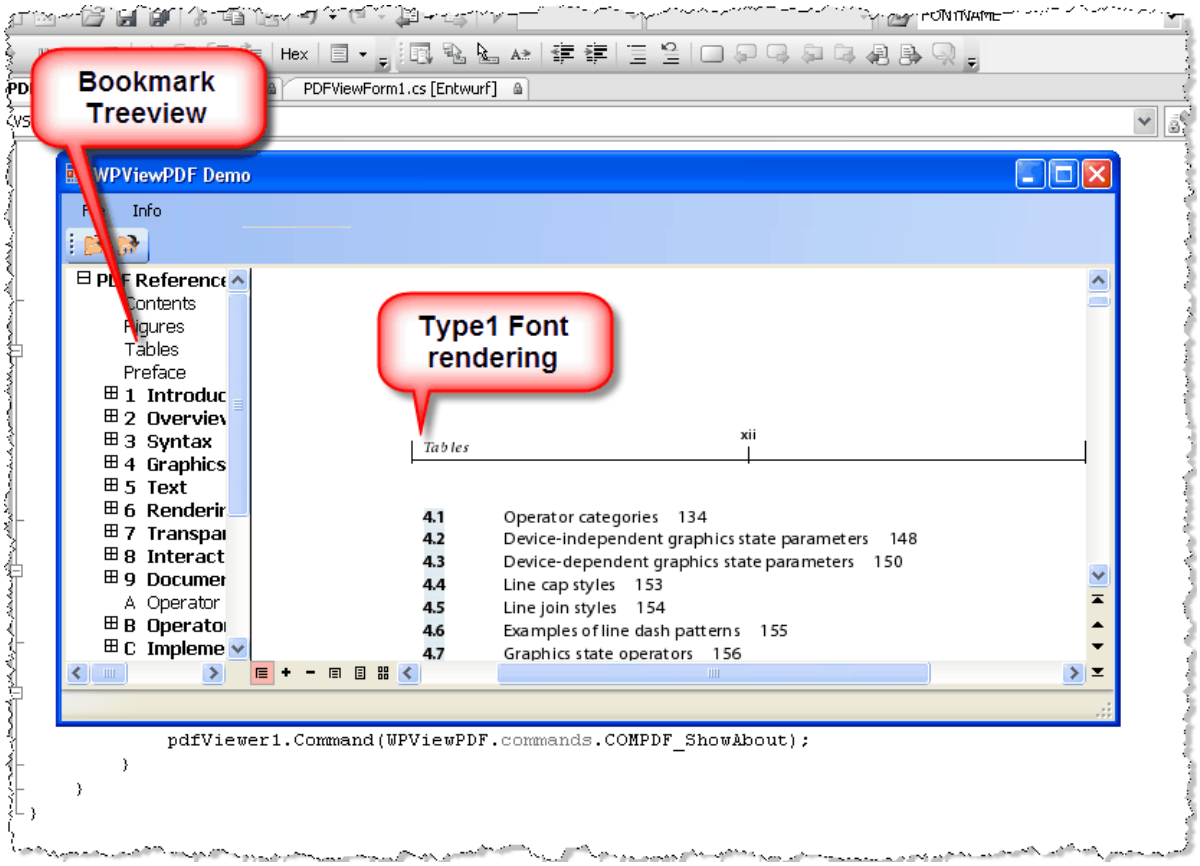
product **WPViewPDF V3** may be interesting for you.
It is not only usable with .NET and as ActiveX, but also in Delphi and C++Builder.

This component was created to view PDF files which were created using the wPDF engine, but it is also capable of viewing PDF files created with other windows based engines which utilize TrueType(tm) fonts. In contrast to many competing solutions WPViewPDF is very fast. You can also use WPViewPDF to convert PDF to EMF and to convert PDF to JPEG.

As "PLUS" edition WPViewPDF is a component which allows you to **view**, **merge, stamp** and **print** PDF files. It is possible to delete certain pages or change the page order.

- integrates font rendering engine to display and print Type 1 fonts
- support for TrueType and Type1 subset fonts.
- support for Type3 fonts
- support for CMYK images.
- new auto scroll mode activated with middle mouse button.
- display bookmarks of PDF file.
- text selection and copy to clipboard.
- optimized load and render methods to open and display PDF files with thousands of pages instantly.
- save PDF as text in RTF (with images), HTML, ANSI and UNICODE
- save PDF page as image files
- multi threaded viewer
- integrated thumbnail view
- PLUS: add draw objects (rectangles, highlights, images) which are movable
- PLUS: edit fields in-place

WPViewPDF used in Visual Studio:



WPViewPDF PLUS

With the PLUS license you can save the PDF information from WPViewPDF which makes it a versatile pdf conversion software. This means you can load in multiple PDF files and save all pages into a new PDF file (=pdf merge, edit pdf).
Certain pages can be marked to be deleted, they will not be displayed by WPViewPDF. When you save the PDF file this pages will be removed. It is also possible to set new security properties (apply, remove encryption) and set property strings int this pdf conversion tool. The WPViewPDF Demo has the PLUS features enabled, but when a new file is created a red cube will be printed on all pages.

With the "PLUS" version it is possible to add text and vector graphics to certain pages of a PDF file (pdf stamping). Any text will be converted to vectors - this allows it to use special fonts. The graphics will be already visible in the viewer before the PDF data has been updated!

You can also modify acroform fields in code.