Developers Manual and API Reference

Table of Contents

1 Welcome to TextDynamic

Welcome to **TextDynamic**® - a powerful library to edit and create formatted text.

TextDynamic is a royalty free word processing and reporting control which can be embedded into applications to create text in code, to offer the end user WYSIWYG editing features and to convert document formats, such as RTF to HTML or RTF to PDF.

TextDynamic is a visual control. It can be used in WinForms application developed with .NET 1.1 or .NET 2.0 Windows development tools (Delphi 2006, VS2003, VS2005). Using the also provided OCX TextDynamic can be embedded into VB6 applications and used with MS Access® or Visual FoxPro® as editing, reporting or printing component.

Product Page . NET | Product Page OCX | API Reference 94 | Support forum

The component was built using the reliable and widely used Delphi Component WPTools® 5 which is based on entirely new code, developed in 2004 and subsequently refined. It also contains many procedures and architecture especially developed for this DLL, for example the modern toolbar interface.

The user interface of TextDynamic can be localized: A special **resource file used by the DLL not only contains the texts for the actions and dialogs in several languages,** but also the images for buttons.

TextDynamic .NET and TextDynamic OCX (*ActiveX*® control) give you the following:

A universal editor for most development systems:
As an OCX it can be used in many popular development systems. The .NET edition has been recently developed, allows seamless integration and is very stable. While other companies charge for their *ActiveX*® and .NET edition of the same control you get here both in one package.

Comprehensive Word Processing Features
TextDynamic includes, already in the basis edition, support for tables, headers and footers and also images and paragraph style sheets. Of course all the basic word processing features such as different fonts, colors, indents and spacing are possible. Optionally it also supports columns, text boxes (a.k.a. text frames) and foot notes.

(This features are activated in the SetEditorMode 114 function)

Maximum performance when working with documents:
The kernel was created using a powerful, fast and effective development language. The code is highly optimized and widely tested. Using the possibility to create MDI applications without the need to have multiple forms you can create applications which require much less resources (see SimulateMDI demo 46).

Optionally integrated SpellCheck
You do not have to worry about external spell checking components, the required engine has been built right into TextDynamic. When you license the spellcheck you will also receive a dictionary compiler to build your own dictionaries.

Optionally integrated PDF creation
The PDF engine which has been integrated into TextDynamic is one of the most widely used (wPDF). It supports font embedding, PDF/A and security features. Embedded metafile images will be converted to vectors and text, not just bitmaps - to create small PDF files which can be printed at high resolution.

Create e-mails easily
TextDynamic contains a powerful interface called "MAPI 81" which collects the required data for an e-mail (Plain, HTML + attached images) and can also send it using the Mail API. Alternatively you can use the integrated MIME 82 writer to create e-mail data (*. MSG) without the need to create any temporary files.

Mail Merge and optional integrated reporting
The TextDynamic reporting engine 88 works with report templates which are just text and will create documents which can be still edited normally. TextDynamic knows to use the DAO interface and can so be used easily in MS Access. For best flexibility and as alternative to DAO there are powerful events to control the reporting process. If you do not need "bands" in your template, you can use the powerful mail merge feature. Since the fields are are not deleted when the data is inserted, a mail merge 72 form can be used to display the fields in a database while the user browses through it.

Integrated Label creation (preview + print)
You can specify all aspects of a mailing label sheet 76 (size, margins, column count) and TextDynamic will display a perfect preview of the output. In this mode it is still possible to edit the text, for example if the used wants to delete some labels which should not be printed. The labels can be created using mail merge.

Maximum flexibility for configuring the user interface:
The toolbar is configured using a XML script. That script, together with the images for the buttons, is loaded from a special package file - we also include an application to edit that file. This file also contains the XML information to translate the texts in the dialogs.
TextDynamic also supports customizable shaded backgrounds 58 for the toolbars:

. You can also configure several aspects 51 of the editor.

DLL "heaven" (and avoid "DLL hell" i.e. installation of program B breaks program A)

With some other component products it is possible that your program suddenly shows a demo nag screen or does not work at all, just because the user installed a different version of the component. To avoid these version conflicts the OCX wrapper loads the DLL from a fully qualified path - and so avoids a version conflict when a different application uses the same OCX.

Customization of the DLL for certain projects:
We are able to add certain commands (ids) if you need special procedures inside the DLL. Please ask for a free quote.

You need server based document creation?
The component wRTF2PDF / TexDynamic Server 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.
Please see our demo web server www.rtf-net.com.

2 WPViewPDF - a PDF viewer control

To view, print and manipulate PDF files right in your Application our product **WPViewPDF** Version 2 may be interesting for you. It is not only usable with .NET and as ActiveX, but also in Delphi and C++Builder.

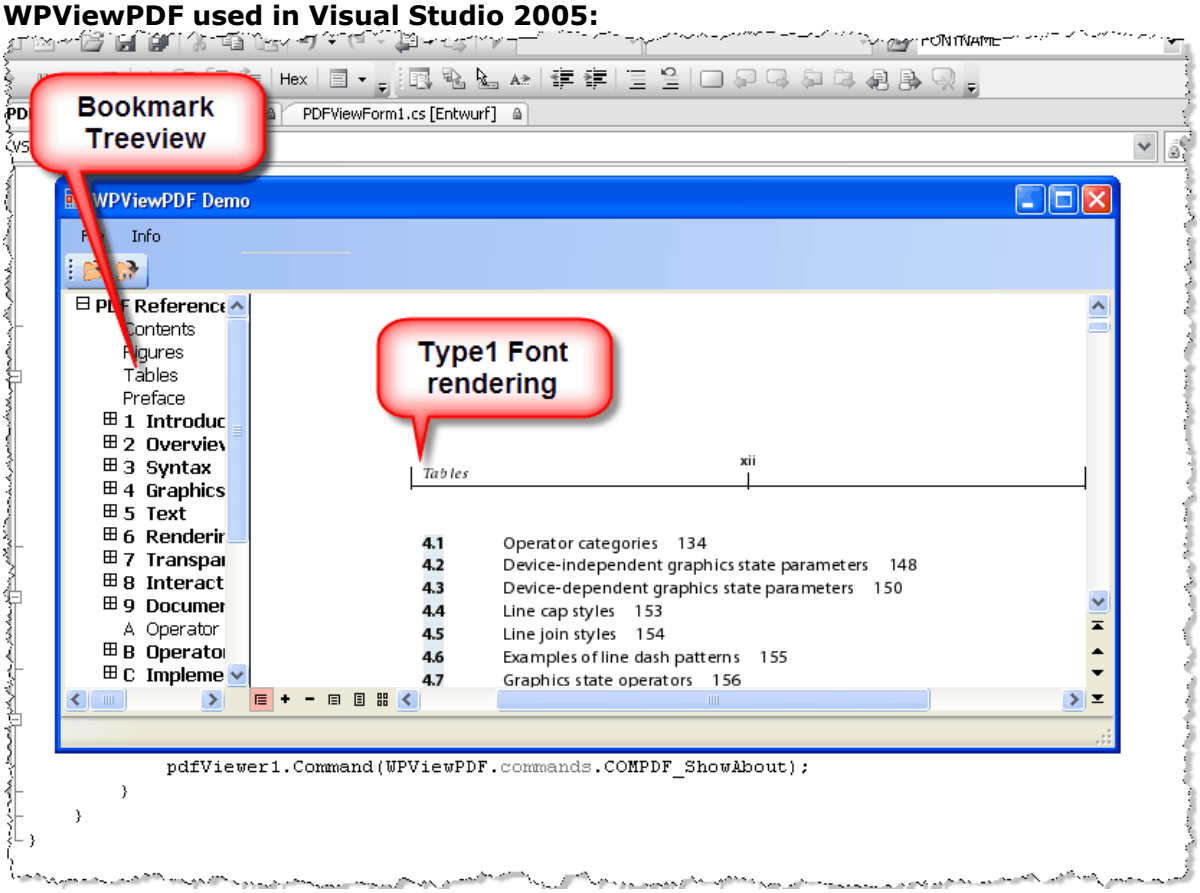
This component was created to view PDF files which were created using the wPDF engine, but it is also capable of viewing PDF files created with other windows based engines which utilize TrueType(tm) fonts. In contrast to many competing solutions WPViewPDF is very fast. You can also use WPViewPDF to convert 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.

New in WPViewPDF V2:

- optionally integrates font rendering engine to display and print Type 1 fonts (simply add 2 DLLs)
- support for TrueType subset fonts.
- improved 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.

WPViewPDF used in Visual Studio 2005:



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 now 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!

3 wPDFControl Documentation

3.1 wPDFControl .NET

3.1.1 RTF to PDF

3.1.1.1 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
```

3.1.1.1 A) WPCOM_SAVE_PAGE_METAFILE = 1311

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

3.1.1.1 B) WPCOM_SELECT_HTML_MODE = 1312

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

3.1.1.1 C) Commands to print text (to printer, not PDF!)

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

When using the COM interfaces 160 you can use this commands 209 to select the printing.

Memo 160.TextCommandStr 209 ID 10 - Select Printer 212
Memo.TextCommandStr ID 11 - Print Text 212
Memo.TextCommandStr ID 12 - Get Printernames 212
Memo.TextCommandStr ID 13 - BeginPrint 212
Memo.TextCommandStr ID 14 - EndPrint 212

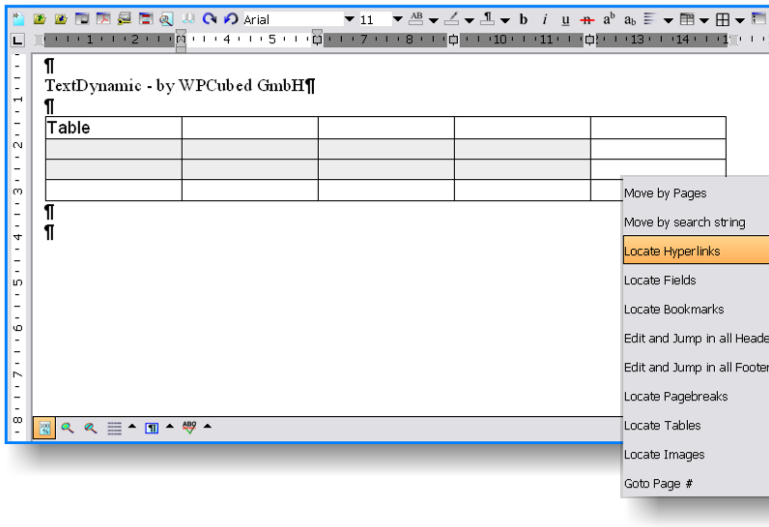
Please note, printing is not thread save!

When you do not use the COM interface use this command ids 5 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
```

4 General

TextDynamic™



This text control is not only a powerful replacement for the .NET richtextbox (richedit DLL) but a complete and full featured word processing component. It can be used with and without its integrated toolbar.

Where you can use TextDynamic™:

- In .NET WinForm applications (as assembly written in C# - DLL).
- In Visual Basic™ 6 applications (as ActiveX™ - OCX)
- In MS Access™ Forms to enter formatted field into databases, mail merge and reporting.
- In Visual FoxPRO™ to enter formatted field into databases, create mailings, print.

What you can do with TextDynamic:

- Process RTF, ANSI, UNICODE and/or HTML texts
- Edit data base memo fields with simple or formatted text
- Prepare and send e-mails using the interface IWPMapi 319.
- Use complete word processing with page layout view, 100% WYSIWYG, header&footer, cascading style sheets (CSS)
- Use integrated spell check
- Work with powerful mail merge 189 (insert and replace formatted text and images)
- Create documents under program control
- Use reporting 372 (= mail merge with bands)
- Use Memo.LabelDef 314 to create, edit and print labels.
- Convert RTF to PDF using the integrated PDF converter 271. The PDF exporter is now now also able to attach data.
- Export HTML to PDF

The central part is the component WPDLLInt 94.

Finally there is an editor component available which is available for a wide range of programming languages (VB6, .NET) and completely customizable. You can use it without the internal toolbars or use XML scripts to create the toolbars according to your needs. TextDynamic provides all state of the art word processor features, including full header/footer support, sections to have different page sizes in one document, tables, nested tables and a multitude of character and paragraph attributes. Optionally footnotes and text boxes (text frames) are supported.

The TextDynamic component package include support for .NET (Framework 1.1 and 2) and also for Access or Visual Basic 6. So you can, at no additional cost, support your legacy applications.

TextDynamic .NET and the OCX work in a very similar way, the interfaces are as interchangeable as possible. But the .NET interface contains some additional features for tight integration, such as the ability to use .NET streams and .NET pictures. So if you have a picturebox with a PNG image loaded, the PNG data can be transferred to the text without any conversion necessary.

But also the OCX has a speciality, using the proerty editor for user defined properties you can load an XML script which configures the editor without the necessity to write a single line of code. This makes it very easy to use the text control in Access since you only have to drop the control and use the property editor.

Since the .NET interface was written in native, managed C# code there is no requirement to ship any OCX with your application. It is not required to modify the registry, simply install the kernel DLL and the assembly DLL into the directory where your dotNET application is installed as well.

Getting Started Topics:

Main Class: WPDLLInt 94, most important are the interfaces Memo 160 and TextCursor 219.

4.1 TextDynamic .NET

This is the perfect replacement for the .NET richtextbox. It can be used to edit database memo fields and also to write letters or documents. It can also be used to combine database records to be printed as one form (mail merge 72, reporting 87).

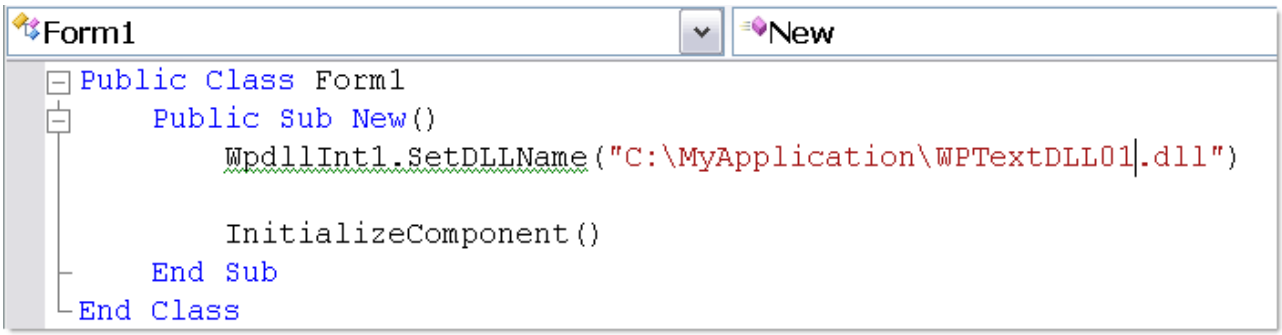
TextDynamic .NET is an assembly developed in C#. It can be compiled with the framework version 1.1 and version 2. This assembly contains the code to access the TextDynamic kernel which has been compiled as native windows code. This way it is extremely fast, for example loading and formatting of the 8MB, 200 pages test file (RTF) took only about 6 seconds.

The path to the kernel DLL is specified using WPDLLInt.SetDLLName(pathname) **before** the initialization of the form.

A good place to do this is the Main() function of the application. The DLL will be loaded when the first instance of the WPDLLInt class in created.

```
[STAThread]
static void Main()
{
        WPDLLInt.SetDLLName("{hkcu}Software\\WPCubed\\TextDynamic\\path");
        Application.Run(new MyDemoApplication());
}
```

If you are using Visual Basic it is best to place the call into the New() method of the form. (The Form.OnLoad event cannot be used since at that time the handle would be already created)



To load the DLL from same direcory as the EXE use this code

Public Sub New()

```
    ' Additional Code - look for DLL in same directory as EXE (bin directory)
    Me.WpdllInt1.SetDLLName(Application.StartupPath & "\WPTextDLL01.DLL")
    ' For Windows Form-Designer
    InitializeComponent()
End Sub
```

It is possible to **read the name from the registry:** Specify the path to the registry key (type string) preceeded by {hkcu} to use HKEY_CURRENT_USER, or {hklm} to use HKEY_LOCAL_MACHINE. In this example the path is specified as reference to a registry entry using {hkcu}. Alternatively you can also sepecify a fully qualified pathname.

Note: The key *Software\WPCubed\TextDynamic\path* is created by the TextDynamic setup script and must not be used in your application. We use it our example projects only.

Please don't mix up the engine DLL with the .NET assembly (WPTDynInt.dll or WPTDynInt2.dll). Only the engine DLL must be specified in SetDLLName!

Please also see the C# sample project 36.

IMPORTANT:

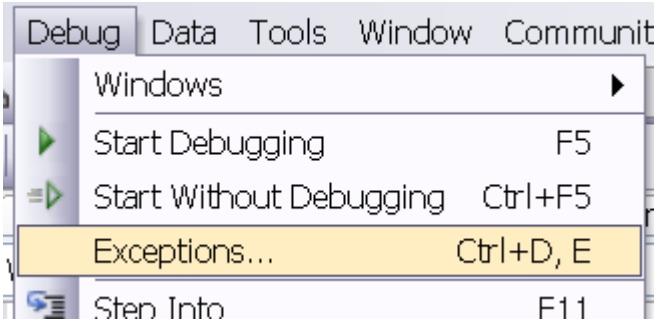
With licensed version please make sure you call wpdllInt1.EditorStart with your license name/key before doing anything else.

```
private void WinForm_Load(object sender, System.EventArgs e)
{
        wpdllInt1.EditorStart( your_name, your_key );
        ....
}
```

4.1.1 Trouble Shooting

I am getting a MDA exception "LoaderLock" when I access the control.

Please open the Menu "Debug", select the item "Exceptions" and in the dialog disable the check at the entry "Loader Lock" listed under "Managed Debugging Assistants".



Where can I set the DLL name in VB.NET

The best place is the New() method of the form. None of the form events is called early enough. Please also note the method SetDLLName 123.

Where should I store the PCC file ?

The gui layout file (buttons.pcc) is searched relatively to the path of the engine DLL. So we recommend to install it into the same directory as te engine DLL - bes is the application directory, not the windows system directory!

I am getting the error "Object Reference not set to an instance of an object."

Please make sure the application is using the correct engine DLL (WPTextDLL01.DLL). Maybe you need to hand copy this to the application directory.

In installed the upgrade 1.30, when I now start my project I an error message:

Please make sure you compiled the project with the new assembly (WPTDynInt.dll or WPTDynInt2.dll). It now uses the strong name technique.
BTW.: The registered version includes the C# code for this assemblies.

When running my .NET application I get the message "Invalid COM object"

In case you use an older version of the .NET interface, please make sure that you do not call ReleaseInt of buffered interface pointers such as Memo.CurrAttr.
The new version automatically makes sure that buffered interfaces are not freed.

I am getting an error message at the end of the application (Sometimes 'Invalid window class name.')
Note: I'm just calling the constructor multiple times and dispose the form container.

a) Please pre-load the DLL using SetDLLName 123.

b) You need to call Form.Dispose(**true**) and not Form.Dispose(false) to make sure the editor instances are also disposed. Otherwise they are never disposed and only the destructor is called when the application closes.

```
public void DisposeForm()
{   Dispose(false);      }
```

My programmatic changes are not saved in a data bound editor

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 219.CheckState 225(10) to do it. This will trigger the PropChange event.

4.2 TextDynamic OCX

This is the perfect replacement for the standard VB richtextbox. It can be used to edit database memo fields and also to write letters or documents. It can also be used to combine database records to be printed as one form (mail merge 72, reporting 87).

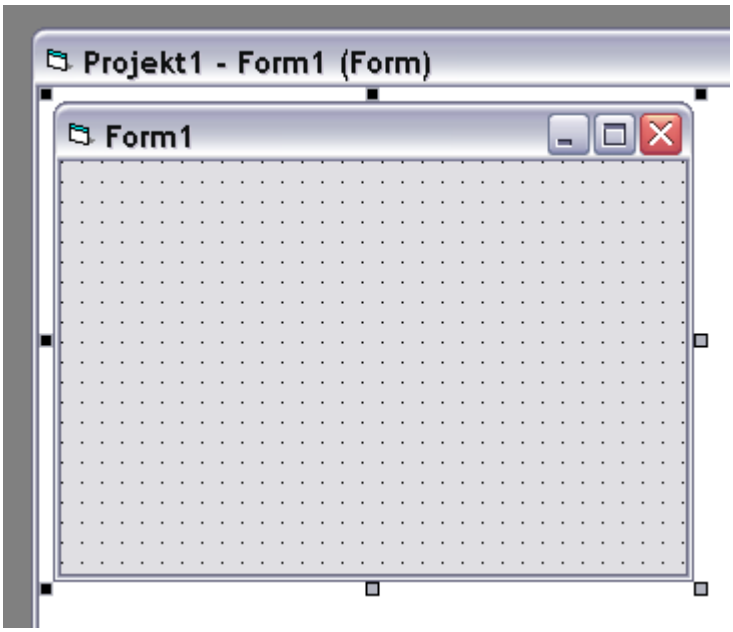
The TextDynamic OCX can be used with ease in MS Access®**.** Without having to write much code you can add a full featured word processor to your data base - you can use it to edit letters, to merge texts, to create PDF files.

In this chapter we describe the few properties which are handled differently in the OCX than in the .NET interface. This are the properties DLLName, InitScripXML, Text, Text2, Readonly and Readonly2.

Please modify the properties **in MS Access only at runtime** - it will save the value only after the control was inserted. (Apparently IPersistPropertyBag interface is not called later)

4.2.1 Step by Step use in VB6

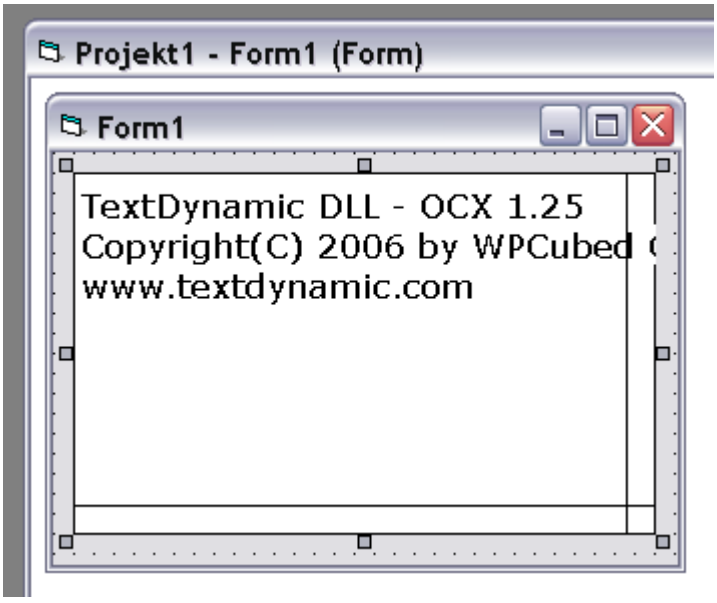
1) create a new project



2) add TextDynamic to the component palette

select "Components..." and then TextDynamic:

3) click on the added component and draw a rectangle on the form to create a control



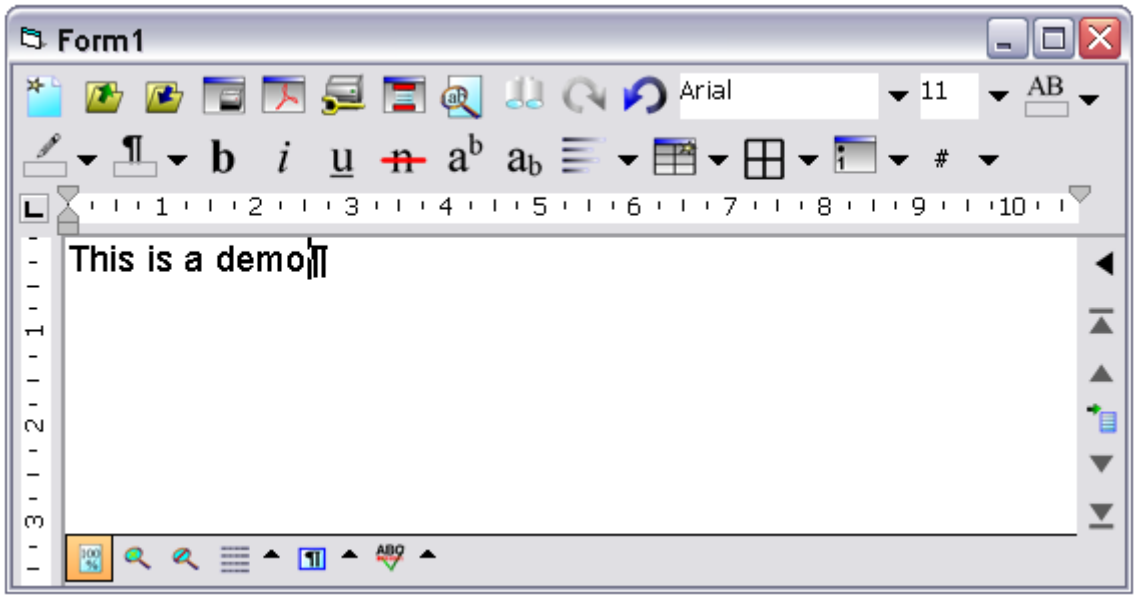
4) now initialize the InitScriptXML property by clicking right on the control and select "Poperties" from the context menu. The popup for click on the button [>>] and select "Create Template" from the list. Close the form.

5) now add code to resize the control dynamically with the parent form

```
Private Sub Form_Resize()
  WPDLLInt1.Left = 0
  WPDLLInt1.Top = 0
  WPDLLInt1.Width = Me.ScaleWidth
```

```
  WPDLLInt1.Height = Me.ScaleHeight
End Sub
```

Start the application and you should get



6) now please add code to tell the control the destination of the engine DLL. This is not required in the demo since it is loaded from the registry, but for an application you distribute you need to modify the property DLLName 14.

7) When you are using the licensed version you need to set your license information using the method EditorStart.

```
Private Sub Form_Load()
 WPDLLInt1.EditorStart( licensename, licensekey )
End Sub
```

8) You can modify the property InitScripXML 15 to change layout of the editor or to load a different PCC file. Instead of using the property InitScriptXML use the method SetLayout 57 and SetEditorMode 53 in the event Form.OnLoad.

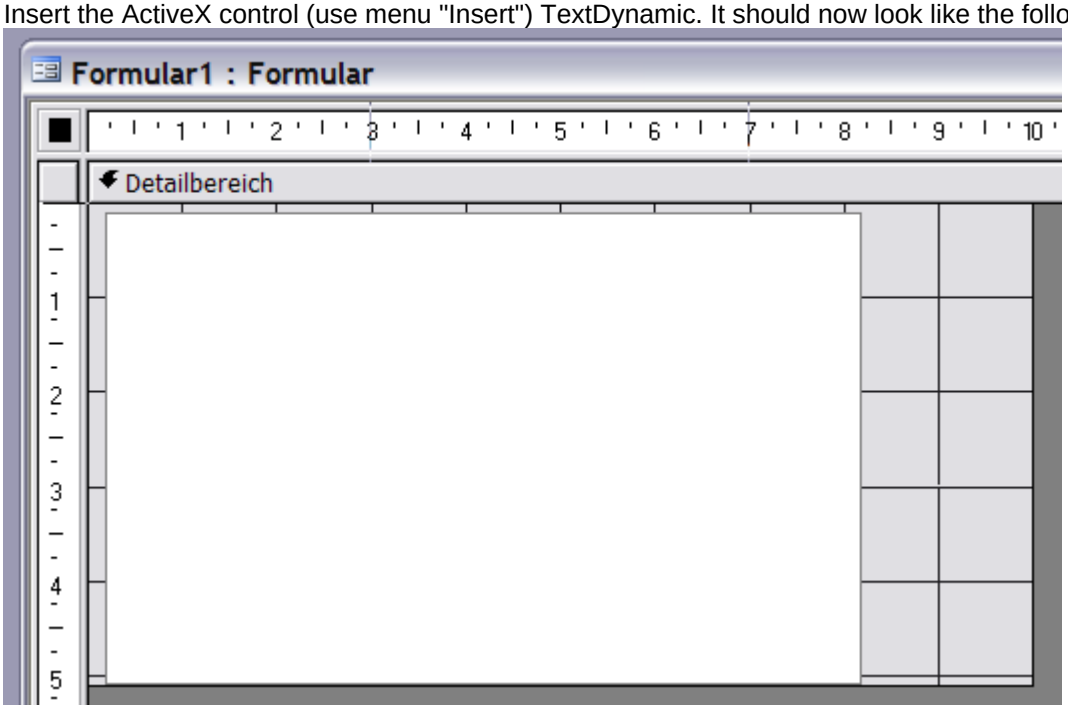
4.2.2 Step by Step use in Access

1) Create an empty database
2) Create a table in this database with text fields and at least one blob field too ho0ld the data of the editor. Please choose the **field type "OleObject"**. The type Memo will not work correctly since it is not only limited to 64KB but also does text conversion which can result in the destruction of binary data. The data used to store the formatted text is plain ASCII code with the binary data of embedded images.

3) Create Form (click on the button  and select the the table we just created for it).

In our example we deactived the scrollbars and the datarecord marker.

Insert the ActiveX control (use menu "Insert") TextDynamic. It should now look like the following screenshot.



4) Now you can add code to initialize the control.

Select the CODE view and add an event handler for the Forrm Event: OnLoad

```
Private Sub Form_Load()
  ' This code requires the respective key to be set in the registry
  ' WPDLLInt0.DLLName = '{hkcu}Software\MyCompany\TextDynamic\path'

  ' Initialize License
  WPDLLInt0.EditorStart "licensename", "licensekey"

  ' load the PCC file
  WPDLLInt0.SetLayout ".\buttons.pcc", "default", "", "main", "main"
  ' Select a single editor with small toolbar
  WPDLLInt0.SetEditorMode 0, 93, 302, 0
  ' Select normal page layout
  WPDLLInt0.SetLayoutMode 0, 0, 100
End Sub
```

Note: The modification of the DLLName and the call to EditorStart is required in an application you are distibuting.

5) As source for the contents select the OleObject field from our table ("LETTER").

This field will be automatically selected for the source for property Text - any updates will be posted there as well.

6) Now add an event to resize the control according to the form:

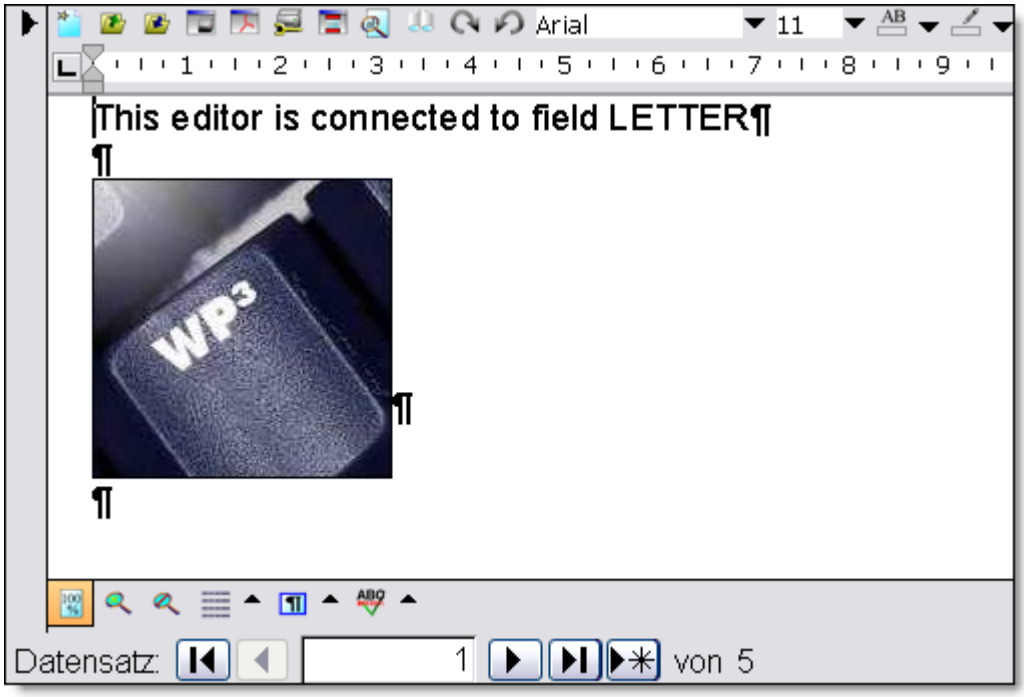
Click on View/Code and add this event handler

```
Private Sub Form_Resize()
  WPDLLInt0.Left = 0
  WPDLLInt0.Width = InsideWidth - 32
  WPDLLInt0.Height = InsideHeight WPDLLInt0.Top - 32
End Sub
```

Close the VB editor and also the form. Save the form as "MainForm"

6) Start the application

You should see a form with a read to use editor which automatically saves and loads the contents from a the binary field in the created table.



7) Add mail merge in MS Access

The template to do mail merge can be either stored in a separate template table or in external files. In any case we need a form to edit the text. Such a form can be created as previously described.

The important change is added code which inserts fields which can be later filled with data loaded from other tables in the project.

4.2.3 Tip for using TextDynamic in Visual FoxPro

Since Version 1.29 all internal interfaces are "dual". This should solve any problems which exists before.

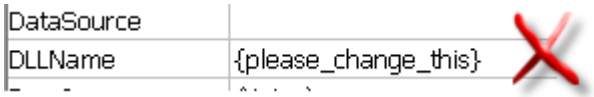
But in case you want to use an IUnknown instead of a IDispatch interface in Visual FoxPro, you can create an object variable and assign the interface you need to use using the method GETINTERFACE:

Example:
ospell=GETINTERFACE(this.SpellCtrl, "iwpspell", "WPTDynInt.ocx")

4.2.4 property DLLName

TextDynamic OCX is an *ActiveX*® control. It contains the code to access the TextDynamic kernel. As usual this OCX needs to be registered as OLE control. Our demo version and the full version uses the same OCX control so you do not have to worry about changed GUIDs. There is also no danger that a newer version of the OCX breakes your application - something you might have experienced with other OCX products because the editing engine itself is loaded from the "TextDynamic Kernel" DLL. This DLL should not be installed to the system directory, it should be installed in the same directory as your application.

When using the OCX, the path of the DLL must be specified in property **DLLName**.



By default the string {please_change_this} is used - that tells the OCX to load the DLL which was installed by the TextDynamic setup script. You need to change that property! It is possible to change this property when the form has been already loaded. Please make sure you use the same setting in all the editor boxes you use in your application. So using a registry reference is the best way to do it.

Please also see the VB6 sample project 20.

Once the name was applied it should not be changed anymore. It is possible to **read the name from the registry:** Specify the path to the registry key (type string) preceeded by {hkcu} to use HKEY_CURRENT_USER, or {hklm} to use HKEY_LOCAL_MACHINE.

Example:
`"{hkcu}Software\MyCompany\TextDynamic\path"` will load the name from the string property *path* under *HKEY_CURRENT_USER\Software\MyCompany\TextDynamic*.

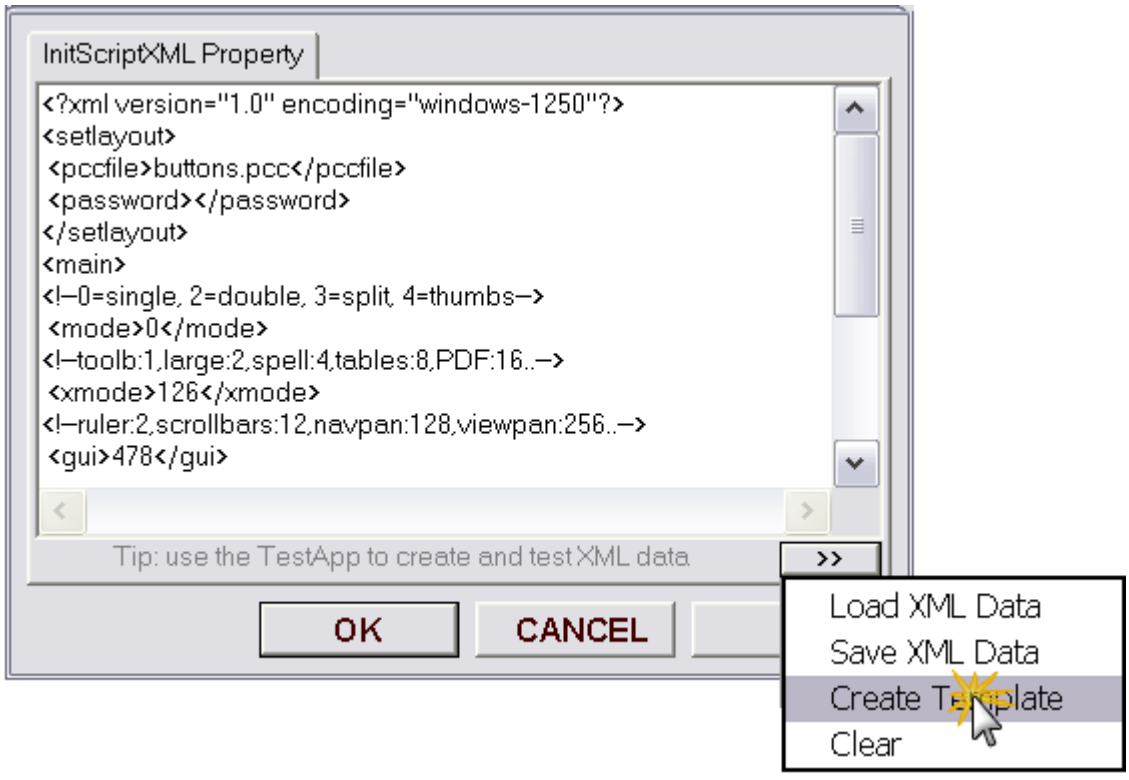
Note: The key *Software\WPCubed\TextDynamic\path* is created by the TextDynamic setup script and must not be used in your application. We use it our example projects only when the property DLLName has its default value "*{please_change_this}*";

Please don't mix up the engine DLL with the .NET assembly (WPTDynInt.dll or WPTDynInt2.dll). Only the engine DLL must be specified in DLLName!

4.2.5 property InitScriptXML

This property is not browsable - this means it is not displayed in the object inspector of VisualBasic. When using Access we recommed to assign the properties in code.

You can assign text to this property in code or you can use the **property editor for custom data.** This editor can be displayed when you click with right mouse button on the control:

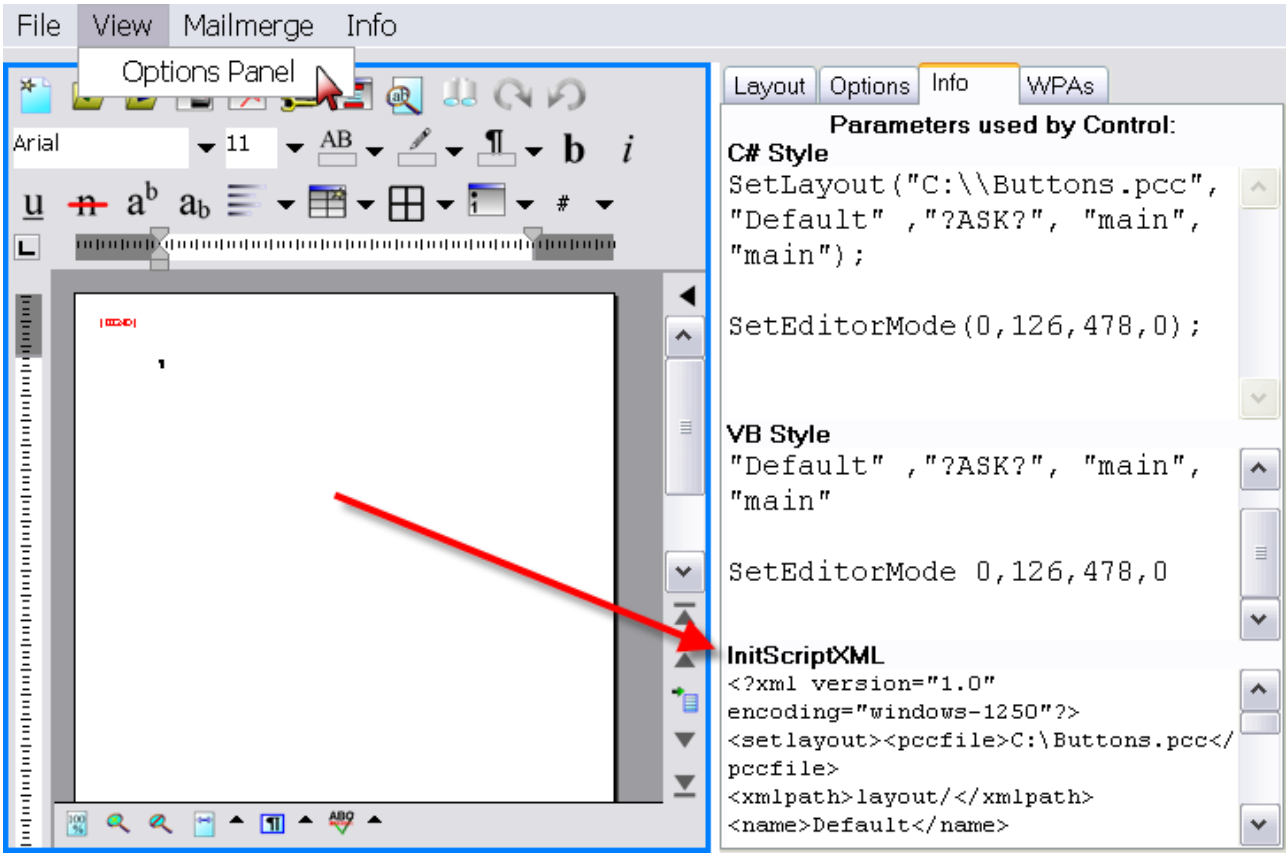


This XML script contains elements to set the editor mode, the extra mode flags, the GUI flags for both editors (main and second). It can also contain the parameters which othewise have to be passed to the procedure SetLayout.

If you use nthe drop down menu of the button [>>] you can load and save the script and also create a default

template.

You can use the test application to create and test XML scrips, see the tab "Info" on the options panel.



4.2.6 property Text

This property is used to bind the control to a database. The format used can be changed using the property TextFormat. You can specify RTF, WPTOOLS, HTML or ANSI. The data used to store the formatted text is plain ASCII code with the binary data of embedded images. To avoid binary data you can use a format string such as "RTF-nobinary", but that will work less efficient since the string becomes larger and loading and saving takes longer.

The property Text2 is used to save and load the data from the second editor - if a second editor is contained inside the control.

The datatype of property Text and Text2 is VARIANT - you can assign a usual unicode string and a variant byte array. When reading this propetry always(*) a variant byte array will be created. To get a standard string from the editor use the method **GetText**. To get a variant array of bytes you can use **GetTextVar**. To load/ insert from a string or a variant array use **SetTextVar**. (The method SetText is similar but will only accept strings)

When binding the editor to a database use a binary field to hold the data. The field should be an image field in SQL server, or an oleobject field in MS Access.

*) If you append "-useolestring," to the property TextFormat, a standard double byte OLE string instead of the variant array of byte will be created.

4.2.7 property Readonly

If this property is true the text in the main editor will be protected. Use the hidden property "Readonly2" to protected in the second editor.

If you need to detect changes in the editor youz can use the event OnChangingText - this event makes it also possible abort a change. The event OnChangedText is triggered after the change.

4.2.8 Trouble Shooting

My Application starts but I cannot type anything

Please initialize the license informatioin correctly - use method EditorStart

Where can I set the DLL in my VB (or similar) project which uses the OCX

You can set the DLL name even when the control is already displayed, the DLL can be exchanged dynamically. However we recomend to set the DLL name in the OnLoad event of the form.

Where should I store the PCC file?

The gui layout file (buttons.pcc) is searched relatively to the path of the engine DLL. So we recommend to install it into the same directory as the engine DLL - bes is the application directory, not the windows system directory!

When I drop the control on the form, why does it display just information text?

The OCX delays loading of the engine DLL to the last possible time. This has been made so to avoid that the DLL is loaded from an undefined or unwanted location. So the DLL is loaded not before the property DLLName has been assigned. This can happen during loading of the form at application start or by your own code. When the control is dropped on the form the property is not touched so the DLL not loaded.

When loading my form I get a "control not found" error message inside of MS Access.

Please open the VB Code editor and select  "References" from menu "Extras". There you can disable the checkmark at the reference which was not found.

When I check out the contents of the blob field using a standard memo, I see only undefined characters

TextDynamic will usually save the text as single byte ASCII text - binary data may be included if images are used. A memo field expects double byte ole string. Such a string will be only saved if property TextFormat contains the parameter "-useolestring', otherwise the property Text will create a varaiant array of bytes.

How to get rid of the P (reversed) symbol at the end of each paragraph?

Please use

```
WPDLLINT1.Memo.SetBProp 197 wpVieOptions,wpShowCRButNotInCells, -1
```

4.3 Dynamic GUI

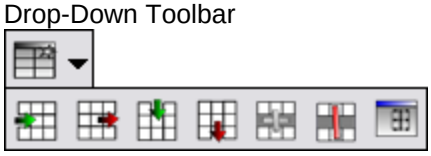
The toolbar plays an important role in an application. It is usually the first thing the user sees and has a high influence on the impression the application makes.

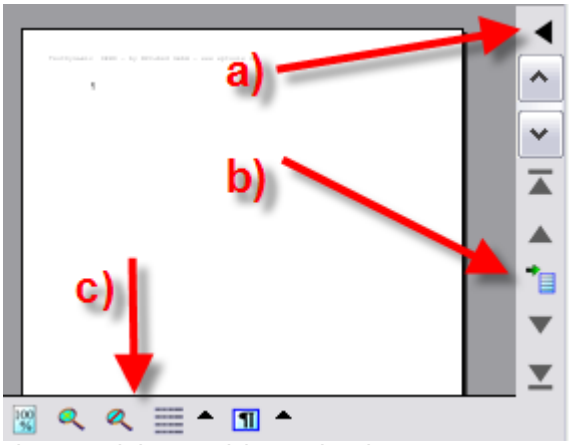
Of course it is possible to use your own toolbar with TextDynamic - we have provided a powerful event (OnUpdateGUI 63) to update the enable/pressed states of the buttons in the most efficient way, but since we know that it is important to have a solution fast, TextDynamic includes its own toolbar as well.

This main toolbar can be fully configured, which does only include the selection and order of the buttons, but also the changing of the images displayed inside the buttons. Like popular word processing applications TextDynamic has also additional tool panels which are located in the lower left, upper right and lower right corner. This smaller toolbars can be configured as well. Also drop down menus and drop down toolbars are supported by the concept.

Main Toolbar (above of ruler):

Drop-Down Toolbar





a) upper right panel ("PanelV1")
b) lower right panel ("PanelV2")
c) lower left Panel ("PanelH1")

Designing such a user interface can be painful - but not with TextDynamic. The user interface is loaded from a separate "PCC" file. This file is created by the "WPCubed Packagefile Manager 424". We include a sample file which should solve most needs. It is easy to add and remove certain buttons. The included PCC file (buttons.PCC) already contains more than 100 modern glyphs in the size 24x24 and 16x16 pixels.

We use XML to initialize the toolbars - the PCC file includes one ore more "editlayout" scripts. The script contains the descrption of buttons and drop down elements which are connected to certain "wpa" actions (see wpa-list 419)

The PCC file can be protected using a password. The password is provided encrypted to the control to allow the PCC file to be loaded. If the password is lost, the PCC file cannot be opened anymore. (The demo version of TextDynamic does not use passwords.)

The user interface has to be initialized using the method **SetLayout** 57. This method will load a package file.

Please note - it is not allowed to distribute the Packagefile-editor to end users. It may only be used by licensed developers.

Please also see the FAQ: Configure the Editor 51

5 Getting Started

5.1 EditorStart

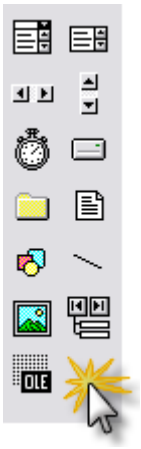
IMPORTANT:
With licensed version please make sure you call wpdllInt1.EditorStart 110 with your license name/ key before doing anything else. The license information is confidential and must not be made visible in the application.

EditorStart should be called before SetEditorMode 114.

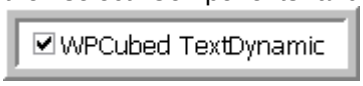
5.2 Install in Toolbox

5.2.1 VB6

To make the control  available in Visual Basic 6, please click on the tool palette with the right mouse button:



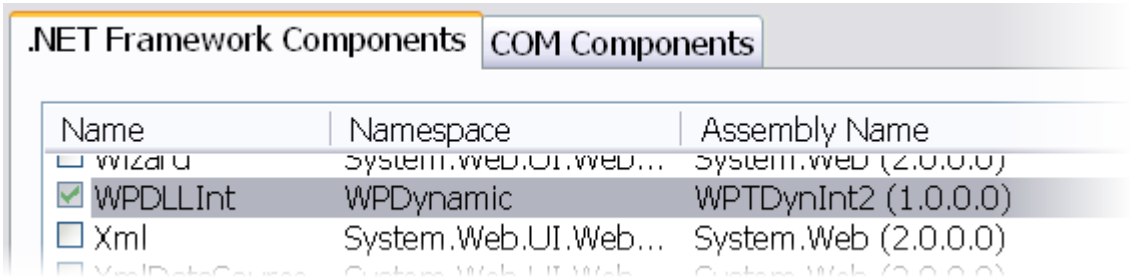
then select "Components" and from the list of installed *ActiveX*® controls choose



Please see the VB6 - First Application 20 and SetEditMode - Example VB6 57

5.2.2 .NET IDE

To install the .NET wrapper please add the DLL WPTDynamic.DLL to the library/toolbox.



In VisualStudio you can also use drag&drop to add the assemblies to the toolbox.

The assembly **WPTDynInt.DLL** was compiled with .NET Framework 1.1, **WPTDynInt2.DLL** was compiled with the .NET Framework 2.

The registered version includes the source code of the interface DLLs - written in C#. So it is possible for you to recompile the assemblies.

5.3 TextDynamic Sample Projects

Below we have collected some examples to make getting started easier.

The demo "Simulated MDI 46" demonstrates the ability to store several documents in one editor control. It will be a good start to explore TextDynamic. In directory Demos\NET1\SimMDI you will find a C# project for Delphi 2006 and for VS2003, in Demos\NET1\SimMDI_VB is a VB.NET project written using VS 2003. In directory Demos\NET2\CS\SimMDI is a C# demo for VS 2005.

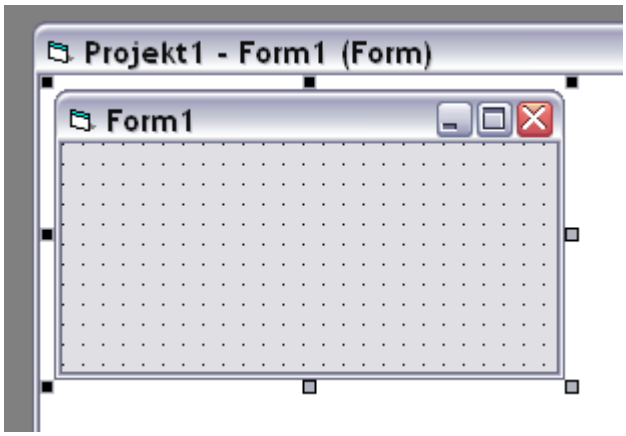
A larger VB.NET example developed using VB.NET 2005 show how to create a MDI application 28 - with multiple child forms which host one TextDynamic control each. This application uses a seperate toolbar which replaces the internal toolbar and shows how the update works.

Please also see "Reporting 89", there we have described how to create an application in **MS Access**. Here the optional reporting feature is used to create a list from record in a data base.

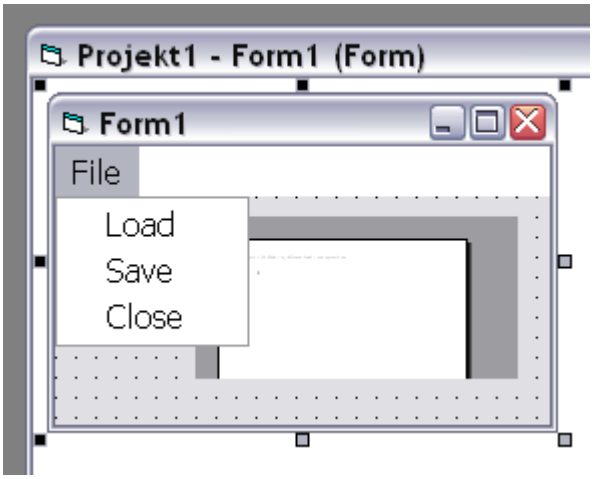
In the capter "Mail Merge 72" we describe how to create mailing letters and now labels!

5.3.1 VB6 - First Application

a) Start a new application



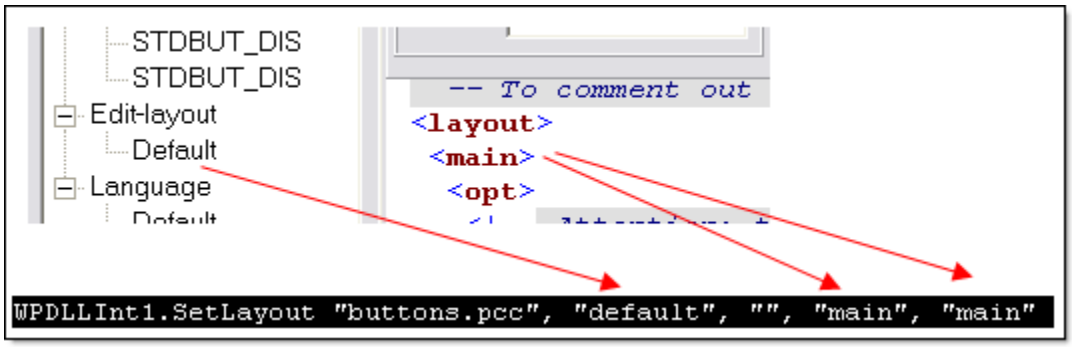
b) add a menu and a WPDLLInt1 object (how to install 18)



c) add an event handler to initialize the editor. Please see the description about SetEditorMode 53 and SetLayout 57.

```
Private Sub Form_Load()
  WPDLLInt1.SetEditorMode 0, 2, 2 + 4 + 8 + 16 + 64 + 128 + 256, 0

  WPDLLInt1.SetLayout "buttons.pcc", "default", "", "main", "main"

  WPDLLInt1.Memo.AutoZoom = wpAutoZoomOff
End Sub
```

In case no toolbar is displayed when you start the application it is most likely that the PCC file has not been found. In this case the application wll beep when started (MessageBeep API) .
If this is not the case and you have selected the propper bits for the GUI mode please check the parameters, and compare to the XML tags used in the WPCubed PackageFile Manager 424:



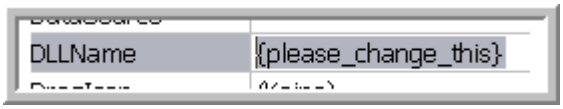
d) add an event handler to resize the editor to fill the window

```
Private Sub Form_Resize()
  WPDLLInt1.Left = 0
  WPDLLInt1.Top = 0
  WPDLLInt1.Width = ScaleWidth - 2
  WPDLLInt1.Height = ScaleHeight - 2
End Sub
```

e) We now attach 3 actions to the menu:

```
Private Sub LoadMen_Click()
  WPDLLInt1.wpaProcess "open", ""
End Sub

Private Sub SaveMen_Click()
  WPDLLInt1.wpaProcess "save", ""
End Sub

Private Sub CloseMen_Click()
  ReadyToClose = True
  Unload Me
End Sub
```

This is not required for the demo - but for a released application set the **property DLLName** to tell the engine the location of the DLL.



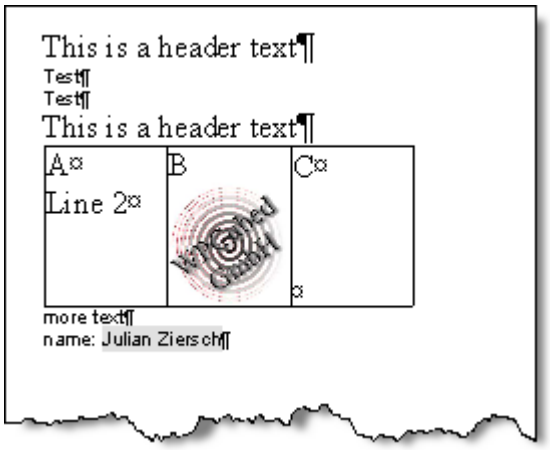
Once the name was applied it should not be changed anymore. It is possible to **read the name from the registry:** Specify the path to the registry key (type string) preceeded by {hkcu} to use HKEY_CURRENT_USER, or {hklm} to use HKEY_LOCAL_MACHINE.

Example:
`"{hkcu}Software\MyCompany\TextDynamic\path"` will load the name from the string property *path* under *HKEY_CURRENT_USER\Software\MyCompany\TextDynamic*.

Note: The key *Software\WPCubed\TextDynamic\path* is created by the WPToolsDLL setup script and must not be used in your application. We use it our example projects only when the property DLLName has its default value "*{please_change_this}*";

5.3.1.1 Create some text under program control

This demo creates a paragraph style "header" and applies to to every other paragraph. It also creates a table.



We first initialize the reference to our working interfaces Memo 160 and TextCursor.

```
Dim Memo As IWPMemo
Dim TextCursor As IWPTextCursor
Set Memo = WPDLLInt1.Memo
Set TextCursor = Memo.TextCursor
```

Now we make sure the mail merge markers, they are usually displayed as << and >> are hidden.

```
WPDLLInt1.SpecialTextAttr(wpInsertpoints).Hidden = True
WPDLLInt1.SpecialTextAttr(wpInsertpoints).CodeTextColor = 0
WPDLLInt1.SpecialTextAttr(wpAutomaticText).BackgroundColor = &HE0E0E0 ' gray
```

And we clear the text.

```
Memo.Clear false, false
```

Now create a text style called 'header'

```
Memo.SelectStyle "header"
Memo.CurrStyleAttr.SetFontface "Times New Roman"
Memo.CurrStyleAttr.SetFontSize 18
```

And create the header and initialize using HTML code.

```
TextCursor.InputHeader 0, "", "<html><center>Testreport <pagenr/> of <pagecount/>"
TextCursor.GotoBody
```

Clear writing attributes - otherwise the style is not used!

```
Memo.CurrAttr.Clear
```

Create the text

```
Memo.CurrPar.StyleName = "header"
TextCursor.InputText "This is a header text"
TextCursor.InputParagraph 0, ""
TextCursor.InputText "Test"
TextCursor.InputParagraph 0, ""
TextCursor.InputText "Test"
TextCursor.InputParagraph 0, "header"
TextCursor.InputText "This is a header text"
```

Start with the table creation - here we use "InputTable 249"

```
TextCursor.InputTable 5000, "AA" ' 50%
TextCursor.InputRowStart 1
TextCursor.InputCell "A", "header"
```

```
TextCursor.InputParagraph 0, "header"
TextCursor.InputText "Line 2"
TextCursor.InputCell "B", "header"
```

Insert an image in a cell

```
TextCursor.InputImage App.Path + "\..\demo.jpg", 0
TextCursor.InputCell "C", "header"
TextCursor.InputRowEnd
```

Now create text after the table

```
TextCursor.InputParagraph 0, ""
TextCursor.InputText "more text"
TextCursor.InputParagraph 0, ""
TextCursor.InputText "name: "
```

And insert a merge field, initialized with a default name

```
TextCursor.InputField "NAME", "Julian Ziersch", False
```

Format the text

```
Memo.Reformat
```

5.3.1.2 Add hotkey Ctrl+B

We want to use CtrlB to toggle the attribute 'bold'

```
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
```

The following values are supported by ToggleStyle and IncludeStyle:

```
  WPWRT_BOLD = 0 ' Bit 1 bold
  WPWRT_ITALIC = 1 ' Bit 2 italic
  WPWRT_UNDERLINE = 2 ' Bit 3 underlined (solod)
  WPWRT_STRIKEOUT = 3 ' Bit 4 strikeout
  WPWRT_SUPERSCRIPT = 4 ' Bit 5 superscript
  WPWRT_SUBSCRIPT = 5 ' Bit 6 subscript
  WPWRT_HIDDEN = 6 ' Bit 7 hidden text
  WPWRT_UPPERCASE = 7 ' Bit 8 all uppercase
  WPWRT_LOWERCASE = 9 ' Bit 10 all lowercase
```

Please use this links to learn more about the features of "TextCursor 219" and "CurrAttr 276 ".

5.3.2 VB.NET - FirstApplication

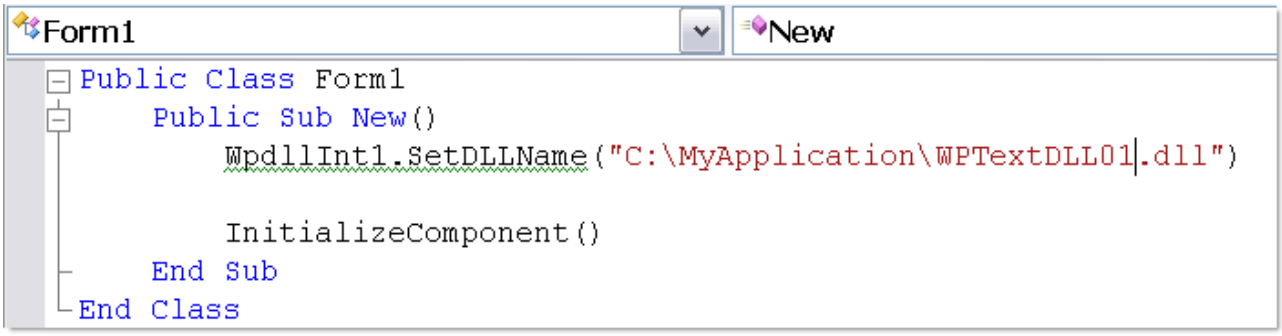
Please also see the demo Create MDI Application (VB.NET) 28 which shows how to use a TToolStrip.

We create a new empty project and add the TextDynamic control. We also add a menu control

and a status bar with 4 sub panels.

```
Private Sub WinForm_Load(ByVal sender As Object, ByVal e As EventArgs)

 Me.wpdllInt1.EditorStart( your_name, your_key );
 Me.wpdllInt1.SetLayout("buttons.pcc", "")
 Me.wpdllInt1.SetEditorMode(EditorMode.wpmodSingleEditor, _
                 (EditorXMode.wpmodexPDFExport Or (EditorXMode.
wpmodexTables _
                 Or (EditorXMode.wpmodexSpellcheck Or EditorXMode.
wpmodexToolbar))),
                 (EditorGUI.wpguiPanelH1 Or _
                 (EditorGUI.wpguiHorzScrollBar Or (EditorGUI.
wpguiVertScrollBar _
                 Or EditorGUI.wpguiRuler))), EditorGUI.wpguiDontSet)

End Sub
```

You can place the code which calls SetDLLName inside the method TForm.New(). The demo does not require this call but if you distribute an application you will need to set a path.



To load the DLL from same directory as the EXE this code is used

```
Public Sub New()
  ' Additional Code - look for DLL in same directory as EXE (bin directory)
  Me.WpdllInt1.SetDLLName(Application.StartupPath & "\WPTextDLL01.DLL")
  ' For Windows Form-Designer
  InitializeComponent()
End Sub
```

Note: You will need to add the WPTextDLL01.DLL and the *.PCC file to the installation script manually.

Please don't mix up the engine DLL with the .NET assembly (WPTDynInt.dll or WPTDynInt2.dll). Only the engine DLL must be specified in SetDLLName!

Now we can already load text. Even the PDF export works. (The demo version includes all options - in the full version options such as spellcheck, PDF, reporting are optional). Table support is always included.

To remove certain elements off the toolbar edit the PCC file using the Package File Manager 424.

a) To update the status bar we add an event handler for OnUpdateGUI 63.

This event receives useful information about the location inside the text + a number of state flags.

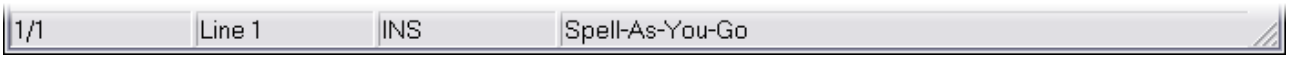
```
Private Sub wpdllInt1_OnUpdateGUI(ByVal Sender As Object,
  ByVal Editor As Integer,
  ByVal UpdateFlags As Integer,
  ByVal StateFlags As Integer,
  ByVal PageNr As Integer,
  ByVal PageCount As Integer,
  ByVal LineNr As Integer)
      Me.stPage.Text = (Convert.ToString(PageNr) & "/"c & Convert.
ToString(PageCount))
      Me.stLine.Text = ("Line " & Convert.ToString(LineNr))
      Me.stIns.Text = IIf(((StateFlags And 2) <> 0), "INS", "")
End Sub
```

b) We want to use the last panel to display the "hint" of the tool button under the mouse cursor.

The OnShowHint event makes it easy:

```
Private Sub wpdllInt1_OnShowHint(ByVal Sender As Object,
  ByVal X As Integer,
  ByVal Y As Integer,
  ByVal Hint As String,
  ByRef Ignore As Boolean)
      Me.stHint.Text = Hint
      Ignore = True
End Sub
```

Now the statusbar shows useful information



We need at least load and save commands in the toolbar.

So we add 2 menu items with this code:

```
Private Sub Loadmenu_Click(ByVal sender As Object, ByVal e As EventArgs)

      Me.wpdllInt1.wpaProcess("open", "")
End Sub

Private Sub SaveMenu_Click(ByVal sender As Object, ByVal e As EventArgs)

      Me.wpdllInt1.wpaProcess("save", "")
End Sub
```

The method wpaProcess takes as parameter the name of a WPA action 433. Using this action names is the easiest possibility to execute standard commands. You can use the test application to test the available actions. Note that the actions are also used inside the XML description for the tool bar.

Note: PDF creation is always available (without license in unregistered mode).
Please try it out using: `Me.wpdllInt1.wpaProcess("DiaExportToPDF","")`
Please also see the interface PDFCreator 271, it contains the properties for the PDF creation.

The GUI definition file of TextDynamic contains the localization of the dialog strings and also action captions and hints. Some action names have been reserved to store the caption for menu drop downs only.

This enum definition can be used for code which creates a menu in code:

```
[Flags]
internal enum wpaMenuCaps
{
	wpaDropDownMenuFile=1,
	wpaDropDownMenuEdit=2,
	wpaDropDownMenuView=4,
	wpaDropDownMenuInsert=8,
	wpaDropDownMenuFormat=16,
	wpaDropDownMenuExtras=32,
	wpaDropDownMenuData=64,
	wpaDropDownMenuTable=128,
	wpaDropDownMenuWindow=256,
	wpaDropDownMenuInfo=512
}
```

Using this action names and a selection of the 'intelligent' actions you can create a framwork to create the menu of your form "on the fly".

We created a menu class which handles the update of the strings completely automatically.

You only need to to implement the **IGetCurrEditor** interface in your winform.

```
Public Class WinForm
	Inherits Form
	Implements IGetCurrEditor
	...

Public Function CurrEdit() As WPDLLInt
	Return Me.wpdllInt1
End Function
```

Now we create the procedure which initializes the menu.

The actions for each menu drop down are provided as string array. This makes this code easy to be updated (green lines).

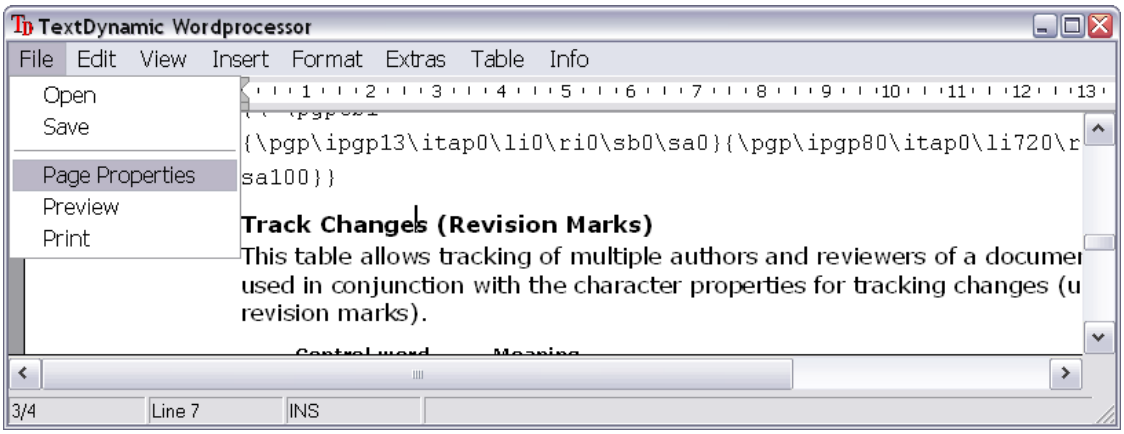
```
Private Sub UpdateMenu(ByVal menu As MainMenu, ByVal selection As wpaMenuCaps)
        Dim textArrayArray1 As String()() = New String()(10  - 1) {}
' FILE
textArrayArray1(0) = New String() { "DiaOpen", "DiaSave", "-", "DiaPagePropEx",
"DiaPreview", "DiaPrint" }
' EDIT
textArrayArray1(1) = New String() { "undo", "redo", "-", "DiaFind", "DiaReplace", "-",
"copy", "paste", "SelAll" }
' VIEW
 textArrayArray1(2) = New String() { "DiaManageHeaderFooter", "ShowCR", "-",
"LayoutNormal", "zoom100", "zoomwidth", "zoomfullpage", "zoomdoublepage" }
' INSERT
 textArrayArray1(3) = New String() { "DiaINSGRAPHIC", "DiaINSSymbol", "DiaINSHyperlink",
"DiaINSFields" }
' FORMAT
textArrayArray1(4) = New String() { "DiaParagraphProp", "DiaParagraphBorder",
"DiaBulletOutlines" }
' EXTRAS
textArrayArray1(5) = New String() { "DiaSpellcheck", "DiaSpellOptions", "SpellAsYouGo" }
' DATA - reserved
        Dim textArray8 As String() = New String(0  - 1) {}
        textArrayArray1(6) = textArray8
' TABLE
        textArrayArray1(7) = New String() { "DiaINSTable", "DiaParagraphBorder",
"InsColBefore", "InsCol", "InsRowBefore", "InsRow" }
```

```
' WINDOW
        Dim textArray10 As String() = New String(0  - 1) {}
        textArrayArray1(8) = textArray10
' INFO
        textArrayArray1(9) = New String() { "DiaWPAbout", "DiaWPDebug" }


Dim textArray1 As String() = New String() { "DropDownMenuFile", "DropDownMenuEdit",
"DropDownMenuView", "DropDownMenuInsert", _
      "DropDownMenuFormat", "DropDownMenuExtras", "DropDownMenuData",
"DropDownMenuTable", "DropDownMenuWindow", "DropDownMenuInfo" }
Dim num1 As Integer = 0
Dim num2 As Integer = 1
Do While (num1 <= 10)
      If ((selection And CType(num2, wpaMenuCaps)) = CType(num2, wpaMenuCaps)) Then
            Dim item2 As New WPAMenuItem(Me, textArray1(num1))
            Dim textArray13 As String() = textArrayArray1(num1)
            Dim num3 As Integer = 0
            Do While (num3 < textArray13.Length)
                  Dim text1 As String = textArray13(num3)
                  Dim item1 As New WPAMenuItem(Me, text1)
                  item2.MenuItems.Add(item1)
                  num3 += 1
            Loop
            item2.Update
            menu.MenuItems.Add(item2)
      End If
      num1 += 1
      num2 = (num2 * 2)
Loop
End Sub
```

After adding to event Form.Load - Initialize MainMenu

```
Me.mainMenu1.MenuItems.Clear  ' Start from scratch!
Me.UpdateMenu(Me.mainMenu1,
(wpaMenuCaps.wpaDropDownMenuInfo Or
wpaMenuCaps.wpaDropDownMenuTable Or
wpaMenuCaps.wpaDropDownMenuExtras Or
wpaMenuCaps.wpaDropDownMenuFormat Or
wpaMenuCaps.wpaDropDownMenuInsert Or
wpaMenuCaps.wpaDropDownMenuView Or
wpaMenuCaps.wpaDropDownMenuEdit Or wpaMenuCaps.wpaDropDownMenuFile)
```

this is how the menu will look at runtime.

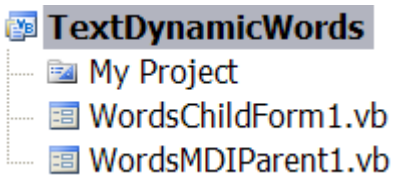


5.3.3 VB.NET - Create MDI Application

5.3.3.1 New Project

The source of this project can be downloaded at http://www.wpcubed.com/tdex/vbnetprj1.zip
We will add further functionality in the future (menus, translation). Suggestions are welcome!

In VisualStudio 2005 we create a new project with a MDI parent form and a regular form.



The IDE will create a template for us - we only need to update the code to use our new child form instead of simply creating a generic form.

```
    Private Sub ShowNewForm(ByVal sender As Object, ByVal e As EventArgs) Handles
NewToolStripMenuItem.Click, NewToolStripButton.Click, NewWindowToolStripMenuItem.Click
        ' Create instance
        Dim ChildForm As New WordsChildForm1
        ' assign parent
        ChildForm.MdiParent = Me

        m_ChildFormNumber += 1
        ChildForm.Text = "TextDynamic Window " & m_ChildFormNumber

        ChildForm.Show()
    End Sub
```

We do not want to discuss the other required changes since the focus of this chapter is how a toolbar can be created and updated and how to work with the menus.

Call SetDLLName

You can place the code which calls SetDLLName inside the method TForm.New(). The demo does not require this call but if you distribute an application you will need to set a path. In a MDI application it is best to call this static method from the main form code.

```
    Public Sub New()
        ' Additional Code - look for DLL in same directory as EXE (bin directory)
        Me.WpdllInt1.SetDLLName(Application.StartupPath & "\WPTextDLL01.DLL")
        ' For Windows Form-Designer
        InitializeComponent()
    End Sub
```

Call EditorStart

WpdllInt1.EditorStart("licensename", "licensekey") has to be called for each editor in the project

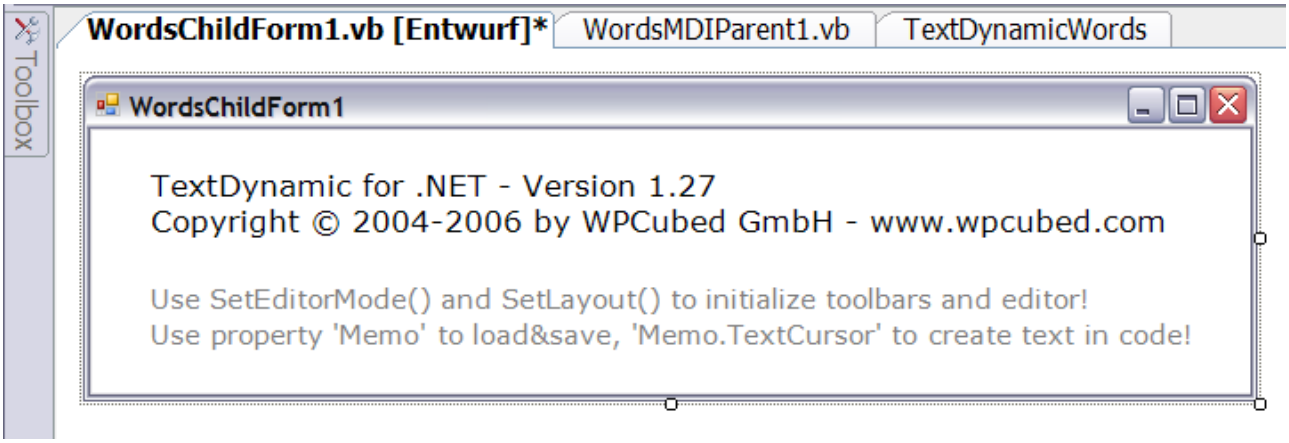
Set License

Please call EditorStart for each TextDynamic editor you use (in Child Form).

5.3.3.2 Create MDI Child window

If TextDynamic was not installed to the IDE already we need to do so. Use the Menu "Extras/External Tools..." and locate the file WPDLLInt2.dll.

Now you can drag the control WPDllInt to the form and resize it to fill the form. Set the property anchors to top, left, right, bottom.



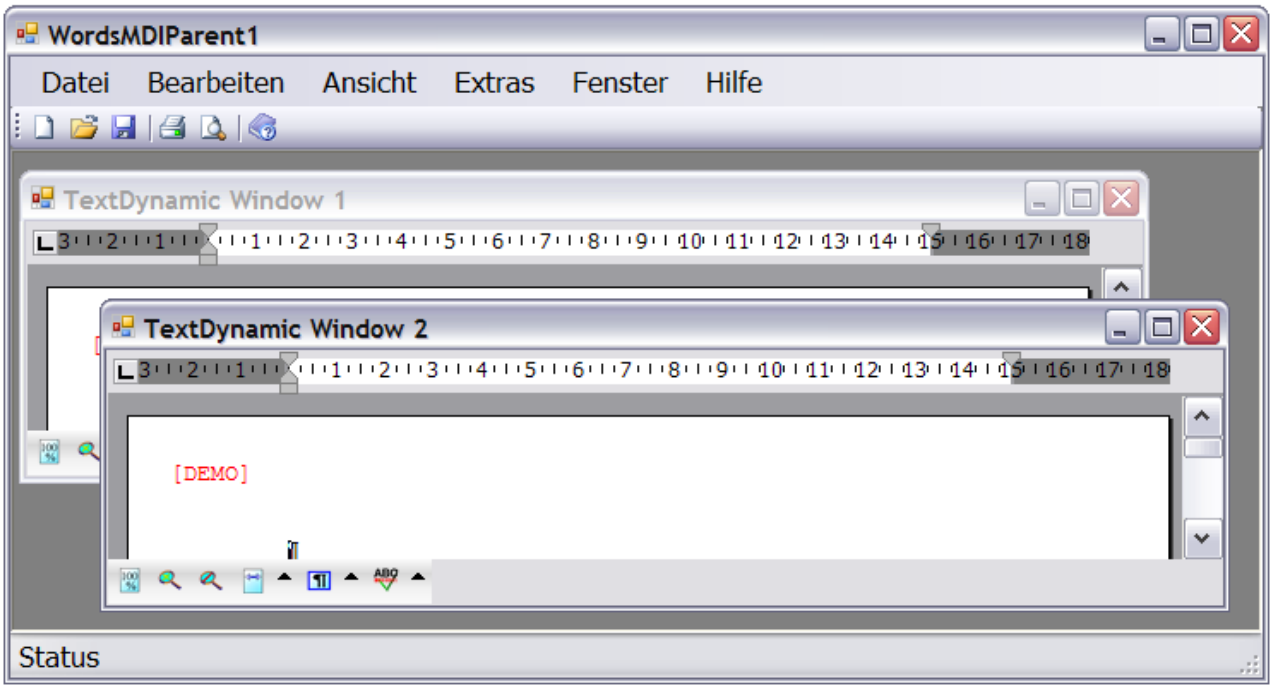
The form now needs some code in the event Form.OnLoad to set the license code and initialize the editor.

```
Private Sub WordsChildForm1_Load(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles MyBase.Load
    ' Set Licenese Code
    WpdllInt1.EditorStart("licensename", "licensekey")
    ' Load the PCC file withot password
    WpdllInt1.SetLayout(".\buttons.pcc", "")
    ' Initialize the editor,  as single editor (no splitscreen)
    ' with 16x16 toolbar use wpmodexToolbar, for 24x24 use wpmodexToolbarLG
    ' but we want to have our own toolbar
    ' with PDF, Spellcheck and table support.
    ' Select the GUI elements ruler, scrollbars and
    ' the lower left panel to change zooming and layout
    ' Since we do not have a second editor window, so use wpguiDontSet
    WpdllInt1.SetEditorMode( _
      WPDynamic.EditorMode.wpmodSingleEditor, _
      WPDynamic.EditorXMode.wpmodexPDFExport Or _
      WPDynamic.EditorXMode.wpmodexSpellcheck Or _
      WPDynamic.EditorXMode.wpmodexTables, _
      WPDynamic.EditorGUI.wpguiRuler Or _
      WPDynamic.EditorGUI.wpguiHorzScrollBar Or _
      WPDynamic.EditorGUI.wpguiVertScrollBar Or _
      WPDynamic.EditorGUI.wpguiPanelH1, _
      WPDynamic.EditorGUI.wpguiDontSet)
End Sub
```

Now start the application.

If the debugger stops in a line without a break point, probably the **MDA loader lock** event is still on. Please open the Menu "Debug", select the item "Exceptions" and in the dialog disable the check at the entry "Loader Lock" listed under "Managed Debugging Assistants".

When the application is now running, please click 2 times non "new". It should look like this (German Edition)



5.3.3.3 Add Font/-Size Selector

Now we add two comboboxes to the toolbar to change the font and font size of the text in TextDynamic.

Both liat can be initailized in the Lod event of the main form.

```
Private Sub WordsMDIParent1_Load(ByVal sender As System.Object,
      ByVal e As System.EventArgs) Handles MyBase.Load
       ' initialize font size combo
       FontSizeSel.Items.Add("6")
       FontSizeSel.Items.Add("7")
       ...
       ' initialize font name combo
       FontNameSel.DropDownStyle = ComboBoxStyle.DropDownList
       Dim aFont As FontFamily
       For Each aFont In FontFamily.Families
          FontNameSel.Items.Add(aFont.Name)
       Next
End Sub
```

To react on a user action we can use the DropDownClosed event. But first we need some code to know which of the child windows is active. The event **MdiChildActivate** can be used to keep track.

```
   ' public variable to represent current TextDynamic editor.
   Public WpdllInt1 As WPDynamic.WPDLLInt

   Private Sub WordsMDIParent1_MdiChildActivate(
                  ByVal sender As System.Object,
                  ByVal e As System.EventArgs) Handles MyBase.MdiChildActivate
      Dim ActiveEditor As WordsChildForm1
      ActiveEditor = TryCast(ActiveMdiChild, WordsChildForm1)
      If ActiveMdiChild Is Nothing Then
```

```
        WpdllInt1 = Nothing
        Text = "TextDynamicWords"
    Else
        WpdllInt1 = ActiveEditor.WpdllInt1
        ' update Caption
        Text = "TextDynamicWords [" + ActiveEditor.Text + "]"
    End If
End Sub
```

Now we can implement the OnDropDownClosed code for the font and font size comboboxes. To update the text attributes it is best to use the **interface** **TextAttr** 276 since it automatically either changes the attributes of selected text (CurrSellAttr) if there is a selection, or the current writing mode (CurrAttr).

5.3.3.3 A) Modify Text

Update font name in editor

```
Private Sub FontNameSel_DropDownClosed(ByVal sender As System.Object,
                ByVal e As System.EventArgs) Handles FontNameSel.DropDownClosed
    If Not WpdllInt1 Is Nothing Then
        WpdllInt1.TextAttr.SetFontface(FontNameSel.Text)
        WpdllInt1.Focus()
    End If
End Sub
```

Update font size in editor

```
Private Sub FontSizeSel_Update(ByVal AndFocus As Boolean)
    If Not WpdllInt1 Is Nothing Then
        Try
        WpdllInt1.TextAttr.SetFontSize(Convert.ToSingle(FontSizeSel.Text))
        If AndFocus Then
                WpdllInt1.Focus()
        End If
        Catch ex As Exception
        End Try
    End If
End Sub

Private Sub FontSizeSel_TextChanged(ByVal sender As System.Object,
        ByVal e As System.EventArgs) Handles FontSizeSel.TextChanged
    FontSizeSel_Update(True)
End Sub

' Update the size if user presses CR in combobox
Private Sub FontSizeSel_KeyPress(ByVal sender As System.Object,
                ByVal e As System.Windows.Forms.KeyPressEventArgs)
                Handles FontSizeSel.KeyPress
    If e.KeyChar = Convert.ToChar(13) Then
        FontSizeSel_Update(False)
    End If
End Sub
```

5.3.3.3 B) Modify Combobox

With the previous code we can change the text in the editor quite easily. But we also want that the combo boxes are updated according to the text at the current cursor position.

The following sub procedure can be used to read font name and font size. If the attribute is not defined in the text the respective combo box will be cleared.
Note: TextDynamic supports undefined attributes, they are important for the operation of paragraph styles. Only text which does not define a certain attribute will use that attribute as

defined in attached paragraph style.

```
Public Sub ReadCurrentAttributes(ByVal MaybeCurrEditor As WPDynamic.WPDLLInt)
   ' We ignore the call if this is not the current editor!
   If MaybeCurrEditor Is WpdllInt1 Then
      ' There is no active editor
      If WpdllInt1 Is Nothing Then
         FontNameSel.SelectedIndex = -1
         FontSizeSel.Text = ""
         FontNameSel.Enabled = False
         FontSizeSel.Enabled = False
      Else ' update the active editor
         FontNameSel.Enabled = True
         FontSizeSel.Enabled = True
         ' Get font name and size from active editor
         Dim ff As String
         If Not WpdllInt1.TextAttr.GetFontface(ff) Then
            FontNameSel.SelectedIndex = -1
         Else
            FontNameSel.SelectedIndex = FontNameSel.Items.IndexOf(ff)
         End If
         Dim s As Single
         If Not WpdllInt1.TextAttr.GetFontSize(s) Then
            FontSizeSel.Text = ""
         Else
            FontSizeSel.Text = Convert.ToString(s)
         End If
      End If
   End If
End Sub
```

But where should be call this procedure?

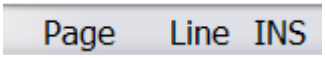
a) At the end of WordsMDIParent1_MdiChildActivate

```
Private Sub WordsMDIParent1_MdiChildActivate(..)
      ...
      ReadCurrentAttributes(WpdllInt1)
End Sub
```

b) In the OnUpdateGUI event of the TextDynamic editor - in the child form add.

```
Private Sub WpdllInt1_OnUpdateGUI(ByVal Sender As System.Object,
                ByVal Editor As System.Int32, ByVal UpdateFlags As System.Int32,
                ByVal StateFlags As System.Int32, ByVal PageNr As System.Int32,
                ByVal PageCount As System.Int32, ByVal LineNr As System.Int32)
                Handles WpdllInt1.OnUpdateGUI
   WordsMDIParent1.ReadCurrentAttributes(WpdllInt1)
End Sub
```

It is common that a word processor display the current position in the status bar. So we add some elements to the StatusStrip control on the main form. The property AutoSize should be off.



Now just three additional lines of code in the above event handler to update the statusbar

```
   WordsMDIParent1.StatusStrip.Items(0).Text = Convert.ToString(PageNr) _
     + "/" + Convert.ToString(PageCount)
   WordsMDIParent1.StatusStrip.Items(1).Text = Convert.ToString(LineNr)
```

5.3.3.4 Add Buttons

Now we add some addtional buttons to the toolbar to set the fonts styles, such as bold, italic.

Into property "Text" for each of this new buttons we enter the name of the wpaAction 419 we intend to execute. For this buttons above this are: bold, italic, underline, left, center, justified, right.

Using the following method we can retrieve the action ID for each of the action names:

```
Dim UpdateLanguage As Boolean = True

Private Sub InitToolbar()
    Dim i, id As Integer
    Dim aName, aCaption, aHint As String
    Dim SelState As Byte()
    Dim tb As ToolStripButton
    If Not WpdllInt1 Is Nothing Then
        SelState = WpdllInt1.wpaGetFlags(0)
        For i = 0 To ToolStrip.Items.Count - 1
            tb = TryCast(ToolStrip.Items(i), ToolStripButton)
            If Not tb Is Nothing Then
                ' Set TAG if still undefined
                If tb.Tag = 0 Then
                    id = WpdllInt1.wpaGetID(tb.Text)
                    If id >= 0 Then
                        tb.Tag = id + 1 ' add one !
                    Else
                        tb.Tag = -1 ' this is not a wpaAction!
                    End If
                Else
                End If
                ' Update if an action
                If tb.Tag > 0 Then
                    ' Update Captions
                    If UpdateLanguage Then
                        If WpdllInt1.Memo.wpaGetCaption(ToolStrip.Items(i).Tag - 1, aName, aCaption, aHint)
Then
                            tb.Text = aCaption
                            tb.ToolTipText = aHint
                        End If
                    End If
                    ' Update Selected and disabled state
                    tb.Enabled = (SelState(tb.Tag - 1) And 1) <> 0
                    tb.Checked = (SelState(tb.Tag - 1) And 2) <> 0
                End If
            End If
        Next
        UpdateLanguage = False ' Update only the first time!
    End If
End Sub
```

This code basically loops all elements in the toolbar and checks the caption of each element if it is a wpaAction or not. If it is an action it updates sets "Tag" = id+1. Then it updates the Text and the states. After this method has been called, the Tag property should bei either -1 or the wpa action ID. This method can be executed again to update the checked and enabled states so we add a "InitToolbar()" at the end of ReadCurrentAttributes!

Now we create one new event for the new buttons (we select the buttons and add the event for all!)

```
' OnClick for all buttons which represent wpaActions
```

```
Private Sub AllButtonClick(ByVal sender As System.Object, ByVal e As System.EventArgs)
                Handles ToolStripButton7.Click ....
   Dim tb As ToolStripButton
   If Not WpdllInt1 Is Nothing Then
      tb = DirectCast(sender, ToolStripButton)
      If tb.Tag >= 0 Then
         If tb.Checked Then
            WpdllInt1.wpaExec(tb.Tag - 1, "0")
         Else
            WpdllInt1.wpaExec(tb.Tag - 1, "1")
         End If
      End If
      InitToolbar()
   End If
End Sub
```

5.3.3.5 Code for the default menu items

OpenFile:

```
 Dim ChildForm As New WordsChildForm1
ChildForm.MdiParent = Me
ChildForm.Text = FileName
ChildForm.Show()
Dim memo As WPDynamic.IWPMemo 160
memo = ChildForm.WpdllInt1.Memo
If (memo Is Nothing) Or _
  Not memo.LoadFromFile(FileName, False, "AUTO") Then
   MessageBox.Show("Cannot load " + FileName)
   ChildForm.Dispose()
End If
```

SaveAs:

```
 If Not WpdllInt1 Is Nothing Then
   Dim memo As WPDynamic.IWPMemo 160
   memo = ChildForm.WpdllInt1.Memo
   If Not memo.SaveToFile(FileName, False, "AUTO") Then
      MessageBox.Show("Cannot write " + FileName)
   End If
End If
```

Cut:

```
If Not WpdllInt1 Is Nothing Then
   Dim memo As WPDynamic.IWPMemo 160
   memo = WpdllInt1.Memo
   memo.CutToClipboard()
End If
```

Copy: memo.CopyToClipboard()

Paste: memo.PasteFromClipboard()

Undo: memo.TextCursor 219.Undo()

Redo: memo.TextCursor.Redo()

SelectAll: memo.TextCursor.SelectAll()

PrintPreview: memo.ShowDialog 216(WPDynamic.DialogID.Preview, "", "")

Print: memo.ShowDialog(WPDynamic.DialogID.Preview, "", "")

Still missing is some code to update the enabled state of the standard buttons and menu items. Since the UpdateGUI event gets a bitfield parameter with the required states we only need a unitilty methiod which processes this bits. We use a public sub procedure in the MDI parent and call it from the UpdateGUI event on the child form.

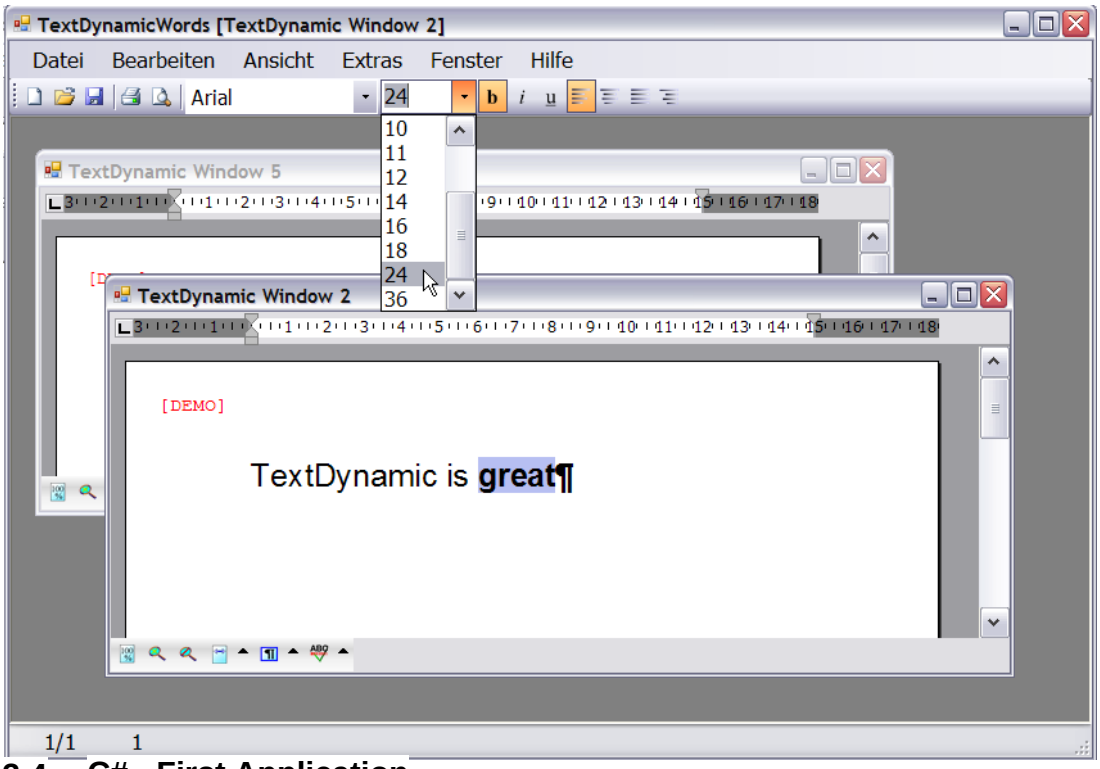
In the MDI Parent:

```
Public Sub UpdateStandardActionState(ByVal StateFlags As Integer)
    ' Modified Flag
    SaveToolStripButton.Enabled = ((StateFlags And 1) <> 0)
    SaveAsToolStripMenuItem.Enabled = ((StateFlags And 1) <> 0)
    ' Inserting mode (update item 3 in status bar)
    If ((StateFlags And 2) <> 0) Then
        StatusStrip.Items(2).Text = "INS"
    Else
        StatusStrip.Items(2).Text = ""
    End If
    ' UNDO State
    UndoToolStripMenuItem.Enabled = ((StateFlags And 4) <> 0)
    ' REDO State
    RedoToolStripMenuItem.Enabled = ((StateFlags And 8) <> 0)
    ' Copy, Cut - if Text is selected
    CopyToolStripMenuItem.Enabled = ((StateFlags And 16) <> 0)
    CutToolStripMenuItem.Enabled = ((StateFlags And 16) <> 0)
End Sub
```

In the MDI Child:

```
Private Sub WpdllInt1_OnUpdateGUI(....) Handles WpdllInt1.OnUpdateGUI
    WordsMDIParent1.ReadCurrentAttributes(WpdllInt1)
    WordsMDIParent1.UpdateStandardActionState(StateFlags)
    WordsMDIParent1.StatusStrip.Items(0).Text = Convert.ToString(PageNr) _
      + "/" + Convert.ToString(PageCount)
    WordsMDIParent1.StatusStrip.Items(1).Text = Convert.ToString(LineNr)
End Sub
```

5.3.3.6 A first start ...



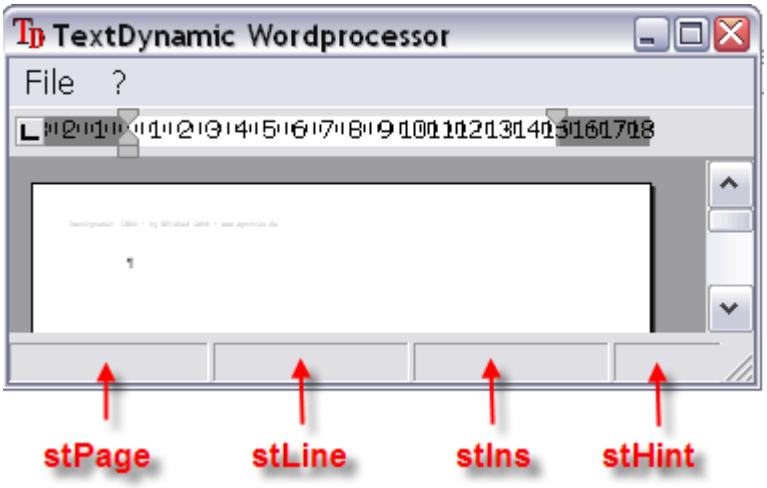
5.3.4 C# - First Application

We create a new empty project and add the TextDynamic control. We also add a menu control and a status bar with 4 sub panels.

Also required is a reference to the WPDynamic namespace:

```
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using WPDynamic; //=namespace for TextDynamic types!
```

When the application runs it will look like

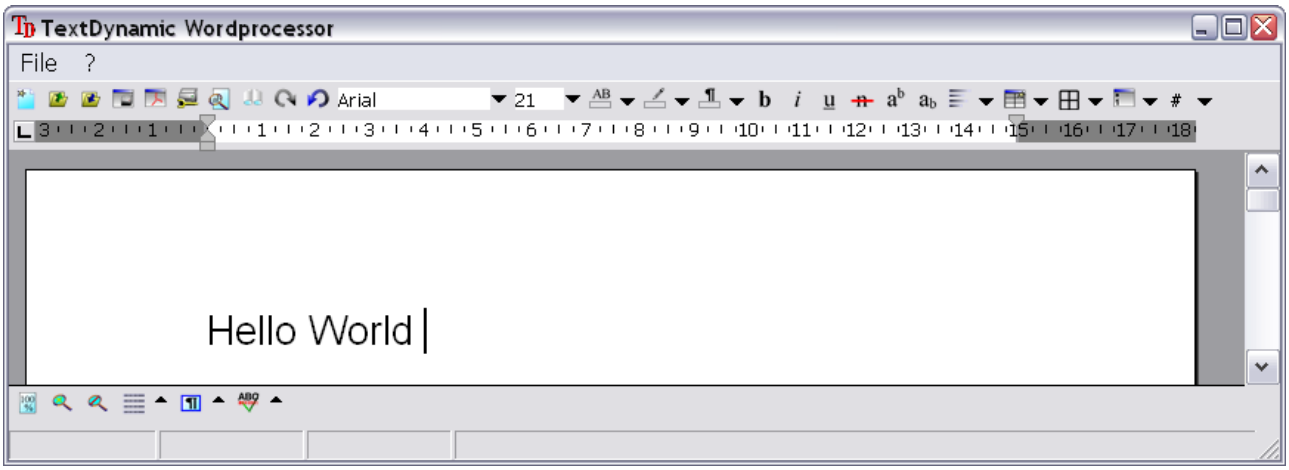


Now we want to add a toolbar. We use the SetLayout method to load the PCC file. We use the standard buttons.PCC file - it is in the same directory as the kernel DLL.

IMPORTANT:
With licensed version please make sure you call wpdllInt1.EditorStart with your license name/key before doing anything else. The license information is confidential and must not be made visible in the application.

```
private void WinForm_Load(object sender, System.EventArgs e)
{
      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|
      // and PDF, Spellcheck and table support.
        EditorXMode.wpmodexPDFExport|
        EditorXMode.wpmodexSpellcheck|
        EditorXMode.wpmodexTables,
      // Select the GUI elements ruler, scrollbars and
      // the lower left panel to change zooming and layout
        EditorGUI.wpguiRuler|
        EditorGUI.wpguiHorzScrollBar|
        EditorGUI.wpguiVertScrollBar|
        EditorGUI.wpguiPanelH1,
      // We have no second editor window, so use wpguiDontSet
        EditorGUI.wpguiDontSet);
}
```

At runtime the application now looks like



Now we can already load text. Even the PDF export works. (The demo version includes all options - in the full version options such as spellcheck, PDF, reporting are optional). Table support is always included.

To remove certain elements off the toolbar edit the PCC file using the Package File Manager 424.

5.3.4.1 Update Statusbar

a) To update the status bar we add an event handler for OnUpdateGUI 143.

This event receives useful information about the location inside the text + a number of state flags.

```
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":"");
}
```

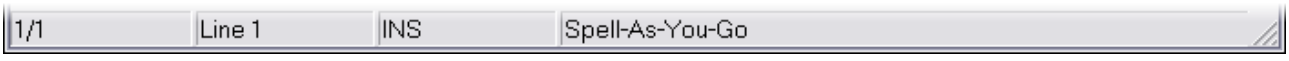
You can also use Memo.Statistic 207 to retrieve information about the count of words, lines and pages.

b) We want to use the last panel to display the "hint" of the tool button under the mouse cursor.

The OnShowHint 142 event makes it easy:

```
private void wpdllInt1_OnShowHint(object Sender, int X, int Y, string
Hint, ref bool Ignore)
{
      stHint.Text = Hint;
      Ignore = true;
}
```

Now the status bar shows useful information



5.3.4.2 Add Menu Commands

We need at least load and save commands in the toolbar.

So we add 2 menu items with this code:

```
private void Loadmenu_Click(object sender, System.EventArgs e)
{
       wpdllInt1.wpaProcess("open","");
}

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

```
        wpdllInt1.wpaProcess("save","");
}
```

The method wpaProcess takes as parameter the name of a WPA action 433. Using this action names is the easiest possibility to execute standard commands. You can use the test application to test the available actions. Note that the actions are also used inside the XML description for the tool bar.

Note: **PDF creation** is always available (without license in unregistered mode).
Please try it out using: `wpdllInt1.wpaProcess("DiaExportToPDF","");`

5.3.4.3 Add Localized Menu Items

The GUI definition file of TextDynamic contains the localization of the dialog strings and also action captions and hints. Some action names have been reserved to store the caption for menu drop downs only.

This enum definition can be used for code which creates a menu in code:

```
[Flags]
internal enum wpaMenuCaps
{
        wpaDropDownMenuFile=1,
        wpaDropDownMenuEdit=2,
        wpaDropDownMenuView=4,
        wpaDropDownMenuInsert=8,
        wpaDropDownMenuFormat=16,
        wpaDropDownMenuExtras=32,
        wpaDropDownMenuData=64,
        wpaDropDownMenuTable=128,
        wpaDropDownMenuWindow=256,
        wpaDropDownMenuInfo=512
}
```

Using this action names and a selection of the 'intelligent' actions you can create a framwork to create the menu of your form "on the fly".

We created a menu class which handles the update of the strings completely automatically.

You only need to to implement the IGetCurrEditor interface in your winform.

```
public class WinForm : System.Windows.Forms.Form,IGetCurrEditor
{
    public WPDynamic.WPDLLInt CurrEdit()  { return wpdllInt1; }
     ....
}
```

Now you can create the procedure which creates the menu. The actions for each menu drop down are provided as string array. This makes this code easy to be updated (green lines).

```
private void UpdateMenu(System.Windows.Forms.MainMenu menu, wpaMenuCaps selection)
{
        // Select the items for the menus - implemented as array of string array
        string[][] wpaMenuItems = new string[10][];
        // DropDownMenuFile
        wpaMenuItems[0]= new string[]
        {"DiaOpen","DiaSave","-","DiaPagePropEx","DiaPreview","DiaPrint"};
        // DropDownMenuEdit
        wpaMenuItems[1]= new string[]
        {"undo","redo","-","DiaFind","DiaReplace","-","copy","paste","SelAll"};
        // DropDownMenuView
        wpaMenuItems[2]= new string[]
```

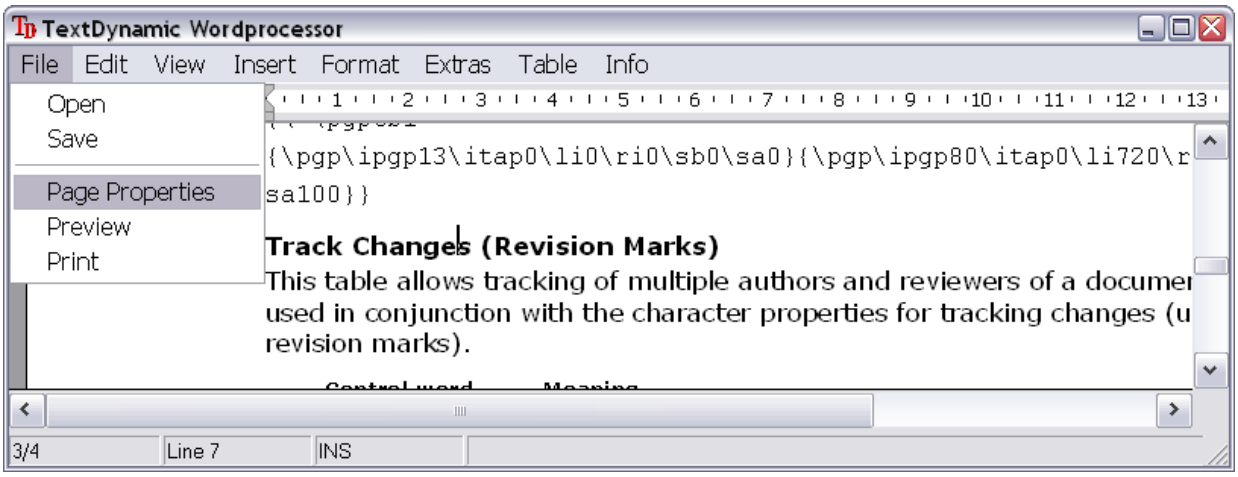
```
        {"DiaManageHeaderFooter","ShowCR","-", "LayoutNormal",
                              "zoom100", "zoomwidth", "zoomfullpage", "zoomdoublepage" };
        // DropDownMenuInsert
        wpaMenuItems[3]= new string[]
        {"DiaINSGRAPHIC", "DiaINSSymbol", "DiaINSHyperlink", "DiaINSFields"};
        // DropDownMenuFormat
        wpaMenuItems[4]= new string[]
        {"DiaParagraphProp", "DiaParagraphBorder", "DiaBulletOutlines"};
        // DropDownMenuExtra
        wpaMenuItems[5]= new string[]
        {"DiaSpellcheck", "DiaSpellOptions", "SpellAsYouGo"};
        // DropDownMenuData
        wpaMenuItems[6]= new string[]
        {/*TODO*/};
        // DropDownMenuTable
        wpaMenuItems[7]= new string[]
        {"DiaINSTable", "DiaParagraphBorder",
               "InsColBefore","InsCol", "InsRowBefore", "InsRow"};
        // DropDownMenuWindow
        wpaMenuItems[8]= new string[]
        {/*TODO*/};
        // DropDownMenuInfo
        wpaMenuItems[9]= new string[]
        {"DiaWPAbout","DiaWPDebug"};
        // ------------------------------------------------------------------
        string[] wpaMenuCapsNames = {
                                      "DropDownMenuFile","DropDownMenuEdit",
                                      "DropDownMenuView","DropDownMenuInsert",
                                      "DropDownMenuFormat","DropDownMenuExtras",
                                      "DropDownMenuData", "DropDownMenuTable",
                                      "DropDownMenuWindow", "DropDownMenuInfo"};
        WPDynamic.WPAMenuItem item, pitem;
        for(int m=0, a=1; m<=10;m++,a=a*2)
        {
                               if(((int)selection&a)==a)
                               {
                                  pitem = new WPDynamic.WPAMenuItem(
                                       this, wpaMenuCapsNames[m]);
                                  // Now create the menu items
                                  foreach(String s in wpaMenuItems[m])
                                  {
                                        item = new WPDynamic.WPAMenuItem(this,s);
                                        pitem.MenuItems.Add(item);
                                  }
                                  pitem.Update();
                                  menu.MenuItems.Add(pitem);
                               }
        }
}
```

After adding to event Form.Load -

```
  // Initialize MainMenu
  mainMenu1.MenuItems.Clear(); // Start from scratch!
  UpdateMenu(mainMenu1,
        wpaMenuCaps.wpaDropDownMenuFile|
        wpaMenuCaps.wpaDropDownMenuEdit|
        wpaMenuCaps.wpaDropDownMenuView|
        wpaMenuCaps.wpaDropDownMenuInsert|
        wpaMenuCaps.wpaDropDownMenuFormat|
        wpaMenuCaps.wpaDropDownMenuExtras|
  //    wpaMenuCaps.wpaDropDownMenuData|
        wpaMenuCaps.wpaDropDownMenuTable|
  //    wpaMenuCaps.wpaDropDownMenuWindow|
```

```
        wpaMenuCaps.wpaDropDownMenuInfo );
```

this is how the menu will look at runtime.



5.3.4.3 A) WPAMenuItem class

```
// This is how the utilty class has been implemented

// Interface required in WinForm
public interface IGetCurrEditor
{
                WPDynamic.WPDLLInt CurrEdit();
}

// Utility class  WPAMenuItem
public class WPAMenuItem : System.Windows.Forms.MenuItem
{
                // the ID
                public int wpaID = -1;
                // the for to work with (must have CurrEdit function)
                private IGetCurrEditor ParentForm;
                // Constructor
                public WPAMenuItem(IGetCurrEditor p, string wpa)
                {
                        ParentForm = p;
                        if(ParentForm.CurrEdit()!=null)
                        {
                                wpaID = ParentForm.CurrEdit().wpaGetID(wpa);
                                if (wpaID<0) Enabled = false;
                                Text = wpa;
                        }
                }
                // Process a click
                protected override void OnClick(EventArgs e)
                {
                   if ((wpaID>=0)&&(ParentForm.CurrEdit()!=null))
                           ParentForm.CurrEdit().wpaExec(wpaID,"");
                }
                // Update the string of this item
                public void Update()
                {
                   string n="",c="",h="";
                   if ((wpaID>=0)&&(ParentForm.CurrEdit()!=null))
            {
```

```
                        if (ParentForm.CurrEdit().Memo.wpaGetCaption(
wpaID, ref n, ref c, ref h))
                        {
                              Text = c;
                        }
                  }
                  for(int i=0;i<MenuItems.Count;i++)
                        if (MenuItems[i] is WPAMenuItem)
                              (MenuItems[i] as WPAMenuItem).Update();
            }
            // Update the current state
            protected override void OnPopup(EventArgs e)
      {
                        if (ParentForm.CurrEdit()!=null)
                        {
                              byte[] flags = ParentForm.CurrEdit().wpaGetFlags(0);
                              if (wpaID>=0)
                {
                                    Enabled = (flags[wpaID]&1)!=0;
                                    Checked = (flags[wpaID]&2)!=0;
                              }
                              for(int i=0;i<MenuItems.Count;i++)
                              if (MenuItems[i] is WPAMenuItem)
                              {
                                    if ((MenuItems[i] as WPAMenuItem).wpaID>=0)
                                    {
                                          MenuItems[i].Enabled =
(flags[(MenuItems[i] as WPAMenuItem).wpaID]&1)!=0;
                                          MenuItems[i]. Checked =
(flags[(MenuItems[i] as WPAMenuItem).wpaID]&2)!=0;
                                    }
                              }
                        }
            }
}
```

5.3.4.3 B) Alternative to WPAMenuItem

The WPAMenuItem class encapsulates some functionality which attaches a "wpa" action to a menu item. You can also use a few lines to replace this code, or to adapt the pricinple to work with other menu or toolbar controls.

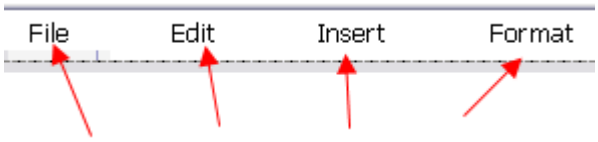
The wpa commands provide an easy way to attach a menu to TextDynamic. The control does not only execute such commands, you can also retieve the caption and hint used for such an action.

Using `WPDLLInt1.CurrMemo.wpaProcess( name, "" )` you can execute the command with the given name.

This makes it, using C# or VB.NET, very easy to create a menu on the fly - but you need to create a class which inherits from System.Windows.Forms.MenuItem and can store an additional value.

```
      internal class TagMenu : System.Windows.Forms.MenuItem
      {
            public int Tag;
      }
```

Now you can implement a function which adds a menu item as submenu of another menu item. (We created 4 menu items in the first level of the main menu which we plan fill)



This is the function which adds the items. It uses the introduced variable 'TagMenu.Tag' to store the id of the action.

```
void AddToMenu(System.Windows.Forms.MenuItem menu, string wpa)
{
   if (wpa=="-") // Separator
   {
                TagMenu newmen = new TagMenu();
                newmen.Text = "-";
                menu.MenuItems.Add( 0, newmen );
   } else
   {
          string caption = wpa;
          string hint = "";
          int wpa_id = WPDLLInt1.Memo.wpaGetID(wpa);
          if(wpa_id>=0)
          {
               WPDLLInt1.Memo.wpaGetCaption(wpa_id,
                         ref wpa, ref caption, ref hint);
               TagMenu newmen = new TagMenu();
               newmen.Text = caption;
               newmen.Tag = wpa_id;
               newmen.Click +=
                 new System.EventHandler(this.wpaMenuClick);
               menu.MenuItems.Add( 0, newmen );
          }
   }
}
```

The function wpaMenuClick can be very short. This is all what is required:

```
private void wpaMenuClick(object sender, System.EventArgs e)
{
       WPDLLInt1.CurrMemo.wpaExec((sender as TagMenu).Tag,"");
}
```

To fill our menu items we use this code - the items are added in reverse order.

```
       // Prepare File menu
              AddToMenu(filemenu, "DiaPreview");
              AddToMenu(filemenu, "PrintDialog");
              AddToMenu(filemenu, "PrinterSetup");
              AddToMenu(filemenu, "-");
              AddToMenu(filemenu, "DiaPageProp");
              AddToMenu(filemenu, "-");
              AddToMenu(filemenu, "SaveAs");
              AddToMenu(filemenu, "Save");
              AddToMenu(filemenu, "Open");
       // Prepare Edit Menu
              AddToMenu(editmenu, "Replace");
              AddToMenu(editmenu, "Search");
              AddToMenu(editmenu, "-");
              AddToMenu(editmenu, "SelAll");
              AddToMenu(editmenu, "Paste");
              AddToMenu(editmenu, "Copy");
              AddToMenu(editmenu, "Cut");
       // Prepare Insert Menu
```

```
            AddToMenu(insertmenu, "DiaINSSymbol");
            AddToMenu(insertmenu, "DiaINSGRAPHIC");
            AddToMenu(insertmenu, "DiaINSTable");
            AddToMenu(insertmenu, "DiaINSHyperlink");
        // Prepare Format Menu
            AddToMenu(formatmenu, "DiaParagraphBorder");
            AddToMenu(formatmenu, "DiaBulletOutlines");
            AddToMenu(formatmenu, "DiaParagraphProp");
```

Similar code to the one above can be used to fill a toolbar as well. Please see the demo "MDI application 28" which shows how to use the ToolStrip control in VB.NET.

5.3.4.4 SetDLLName

This is not required for the demo - but for a released application you need to specify the DLL name using the **static** function **`WPDLLInt1.SetDLLName(string)-`** `this` tells the wrapper the location of the DLL.

This procedure should be called before any text control is created, a good place is the Main() function or, in VB.NET, Form.New().

Once the DLL was loaded the name cannot be changed anymore.

It is possible to **read the name from the registry:** Specify the path to the registry key (type string) preceeded by {hkcu} to use HKEY_CURRENT_USER, or {hklm} to use HKEY_LOCAL_MACHINE.

Example:
`"{hkcu}Software\MyCompany\TextDynamic\path"` will load the name from the string property *path* under *HKEY_CURRENT_USER\Software\MyCompany\TextDynamic*.

```
[STAThread]
static void Main()
{
   WPDLLInt.SetDLLName("{hkcu}Software\\MyCompany\\TextDynamic\\path");
   Application.Run(new WinForm());
}
```

Note: The key *Software\WPCubed\TextDynamic\path* is created by the WPToolsDLL setup script and must not be used in your application. We use this string as default setting inside the C# code.

You can also store the DLL in same directory as the assembly and use this code to load it.

```
// look in same directory as assembly
Assembly TextDynamicAssembly = Assembly.GetAssembly(typeof(WPDLLInt));
string TextDynamicDLL = Path.GetDirectoryName(TextDynamicAssembly.Location) + "\\WPTextDL
// If not existent there look in same directory as executable
if (!File.Exists(TextDynamicDLL))
{
   TextDynamicDLL = Path.GetDirectoryName(Application.ExecutablePath) + "\\WPTextDLL01.dl
}
// Set the DLL name (prelaod the DLL)
WPDynamic.WPDLLInt.SetDLLName(TextDynamicDLL);
```

5.3.5 C# - Second Application

Our second application developed in C# should test all available wpa action names.

We need a form with 3 elements:

Now we add an event handler for the VisibleChanged event of the form. In this event handler we first select the editor mode for the editor and load the layout (PCC file).

Then we initialize the listbox by adding the names and captions of all available wpaActions.

```
using WPDynamic;

private void WinForm_VisibleChanged(object sender, System.EventArgs e)
      {
                // Select Editor Mode !
                WPDLLInt1.SetEditorMode(
                EditorMode.wpmodSingleEditor,
                EditorXMode.wpmodexToolbarLG,
                // EditorGUI.wpguiGutter +
                EditorGUI.wpguiHorzScrollBar |
                EditorGUI.wpguiPanelH1|
                // EditorGUI.pguiPanelH2 |
                EditorGUI.wpguiPanelV1|
                EditorGUI.wpguiPanelV2 |
                EditorGUI.wpguiRuler  |
                EditorGUI.wpguiVertRuler  |
                EditorGUI.wpguiVertScrollBar
                , EditorGUI.wpguiDontSet );

                // Load buttons and change language (if required)
                WPDLLInt1.SetLayout("buttons.pcc","");
                // WPDLLInt1.SetLanguage("DE");

                // This would be NOT allowed before WPDLLInt1.SetEditorMode !
                WPDLLInt1.Memo.TextCursor.InputText("Hello World");

                // Init Listbox
                string n = "",c ="",h ="";
                for(int i=0; i<1000;i++)
                {
                      if (!WPDLLInt1.Memo.wpaGetCaption(i,ref n, ref c, ref h)) break;
                      listBox1.Items.Add(n+"="+c);
                }
      }
```

Now we only need one line of code to be attached to the button:

```
private void button1_Click(object sender, System.EventArgs e)
{
   WPDLLInt1.Memo.wpaExec(listBox1.SelectedIndex, "");
}
```

Now you can start the application, select one of the action entries and click on the execute button.

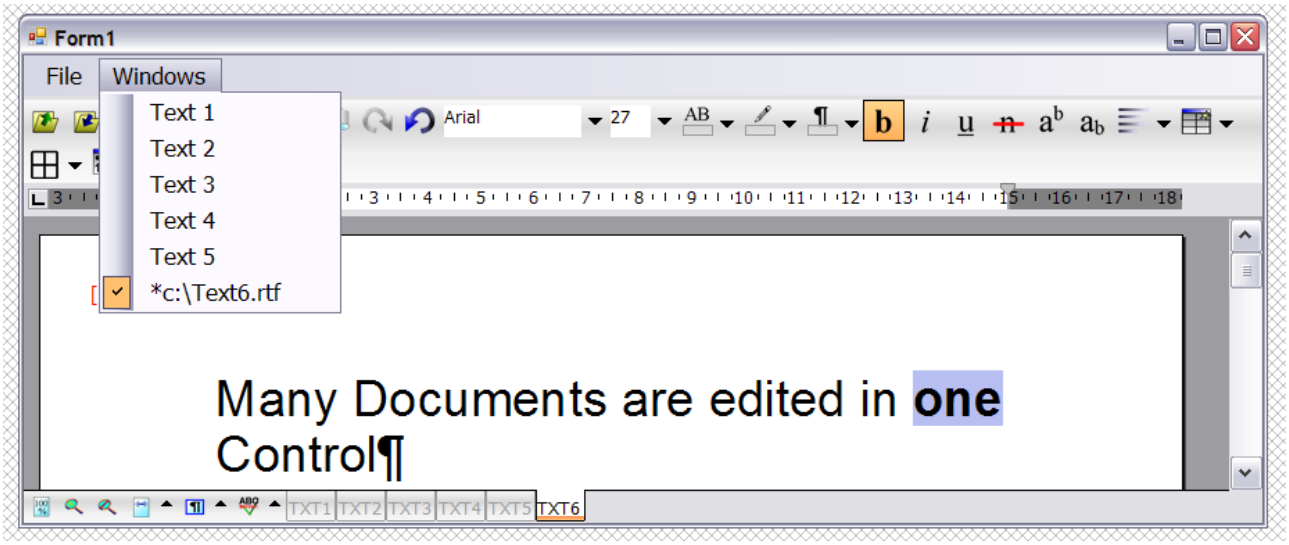
(Note: Some of the actions are not used at present time, some can be only used when used by drop down combos in the toolbar. Please see the WPA-Name List 433.)

5.3.6 Demo Project - Simulate MDI

This demo will be a good start to explore TextDynamic. In directory Demos\NET1\SimMDI you will find a C# project for Delphi 2006 and for VS2003, in Demos\NET1\SimMDI_VB is a VB.NET project written using VS 2003. In directory Demos\NET2\CS\SimMDI is a C# demo for VS 2005.

TextDynamic has the ability to **store several documents in one editor**. If you create a "double editor" you can work with 2 texts in two editor windows. But this is not what is meant here. Here we show how to create many documents and hold them all in the editor. We can switch between them and remove the documents if required. This all works like the MDI example only that just one editor is used.

In the demo directory you will find a C# projects for Delphi 2006 and Visual Studio 2005

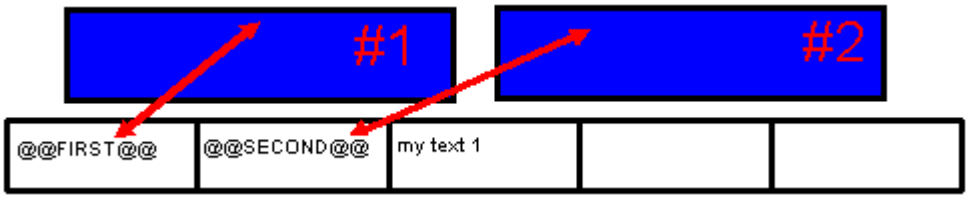


Advantages:
- The zooming state will remain the same for all documents
- Other properties, such as spellcheck mode stays the same
- Resources are saved
- No external toolbar is required
- Undo still works for the individual document
- When switching between editors the previous cursor position is restored

Disadvantages:
- You cannot tile the windows
- It is difficult to display more documents at the same time (using a double editor it is possible to show 2, but keeping track is more complicated)

The simulated MDI demo uses the "RTFData" API. Using this you can create a different RTFData element in

the internal list of TextDynamic, give it a name and a caption (for the tabset) and activate it or delete it.



The RTFData API is defined in interface IWPMemo:

```
void RTFDataAdd(string Name, string Caption) // create a new element
bool RTFDataSelect(string Name) // select an element
void RTFDataDelete(string Name) // delete an element
void RTFDataAppendTo(string Name, bool CreateNewPage) // append the current text to the given
element (maybe create it)
```

Please make sure you select a different one before you delete an element. By default the editor #1 use the element with the name @@*FIRST*@@, the optional editor #2 uses @@*SECOND*@@. You can always "return" to the first element.

RTFDataAdd will first check if it is required to create a new element. If not, it will just set the caption. (The caption can be displayed in the tabset "PanelH2")

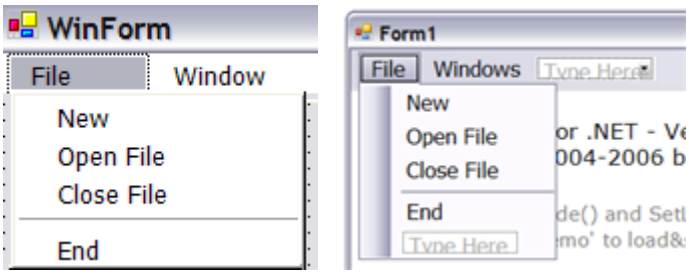
We use method RTFDataAppendTo in the label printing example 76. It is very useful in combination with mail merge since you can create a document from multiple copies of the same text - without the need to activate the editor #2!

In our simulated MDI example we create a simple form, place a TextDynamic control.

The text control is initialized in Form.OnLoad:

```
private void WinForm_Load(object sender, System.EventArgs e)
{
  // wpdllInt1.EditorStart( your_name, your_key );
  // Load the PCC file (no password)
  wpdllInt1.SetLayout(".\\buttons.pcc","");
  // Initialize the editor
  wpdllInt1.SetEditorMode(
  // as single editor
  EditorMode.wpmodSingleEditor ,
  EditorXMode.wpmodexToolbarLG|
  EditorXMode.wpmodexPDFExport|
  EditorXMode.wpmodexSpellcheck|
  EditorXMode.wpmodexTables,
  EditorGUI.wpguiRuler|
  EditorGUI.wpguiHorzScrollBar|
  EditorGUI.wpguiVertScrollBar|
  EditorGUI.wpguiPanelH1,
  EditorGUI.wpguiDontSet);
  // see: http://www.wpcubed.com/manuals/tdref/IDH_WPDLLInt_SetLayoutMode.htm
  wpdllInt1.SetLayoutMode(7,1,100);
  // Set Size - align to complete window
  wpdllInt1.SetBounds(
        ClientRectangle.Left,
        ClientRectangle.Top,
        ClientRectangle.Right-ClientRectangle.Left,
        ClientRectangle.Bottom-ClientRectangle.Top  );
  wpdllInt1.Anchor =
        AnchorStyles.Left|AnchorStyles.Right
        |AnchorStyles.Top|AnchorStyles.Bottom;
}
```

Now we add a main menu with two main items and some items in the first, the File menu. In the .NET 2 example we used a MenuStrip. (Instead of MenuItems use DropDownItems!)



To keep track of the loaded files we can use the item list in the "Window" menu. We could also use a string list or combo box.

We need a new MenuItem class which can store a custom string value, "RTFID".

```
.NET 1.1
public class TagMenuItem : System.Windows.Forms.MenuItem
{
        public string RTFID;
}

.NET 2.0
public class TagMenuItem : System.Windows.Forms.ToolStripMenuItem
{
            public string RTFID;
}
```

This function create a new menu item and add a RTF data element. There is also the OnClick event handler which selects a data element.

```
private int textnr = 0;
private TagMenuItem currentmenu;

private void TagMenuItem_Click(object sender, System.EventArgs e)
{
        wpdllInt1.Memo.RTFDataSelect(((TagMenuItem)sender).RTFID);
        wpdllInt1.Memo.TextCommand(5,0,0);
        currentmenu = ((TagMenuItem)sender);
        for(int i=0;i<WindowMenu.MenuItems.Count;i++)
          WindowMenu.MenuItems[i].Checked = false;
/* .NET 2 code:
       for (int i = 0; i < WindowMenu.DropDownItems.Count; i++)
           ((ToolStripMenuItem)WindowMenu.DropDownItems[i]).Checked = false;
*/
        currentmenu.Checked = true;
}

private TagMenuItem AddANewFile()
{
   TagMenuItem men = new TagMenuItem();
   men.Click += new System.EventHandler(this.TagMenuItem_Click);
   men.Text  = "Text " + Convert.ToString(++textnr);
   men.RTFID = "TXT" + Convert.ToString(textnr);
   WindowMenu.MenuItems.Add(WindowMenu.MenuItems.Count,men);
   //.NET 2: WindowMenu.DropDownItems.Add(men);
   wpdllInt1.Memo.RTFDataAdd(men.RTFID,men.RTFID);
   wpdllInt1.Memo.RTFDataSelect(men.RTFID);
   men.RadioCheck = true;
   for(int i=0;i<WindowMenu.MenuItems.Count;i++)
         WindowMenu.MenuItems[i].Checked = false;
/* .NET 2 code:
       for (int i = 0; i < WindowMenu.DropDownItems.Count; i++)
           ((ToolStripMenuItem)WindowMenu.DropDownItems[i]).Checked = false;
*/
   men.Checked = true;
   currentmenu = men;
   wpdllInt1.Memo.TextCommand(5,0,0);
   CloseFile.Enabled = (WindowMenu.MenuItems.Count>1);
   return men;
}
```

We use this code in the click events:

New:

```
private void NewFile_Click(object sender, System.EventArgs e)
{
        AddANewFile();
}
```

Open:

```
private void OpenFile_Click(object sender, System.EventArgs e)
{
        TagMenuItem old, men;
        old = currentmenu;
        men = AddANewFile();
        if(!wpdllInt1.Memo.Load("","",""))
        {
                if(old!=null)wpdllInt1.Memo.RTFDataSelect(old.RTFID);
                else wpdllInt1.Memo.RTFDataSelect("");
                wpdllInt1.Memo.ReformatAll(false, true);
        }
        else men.Text = wpdllInt1.Memo.LastFileName;
}
```

Close:

```
private bool AbortClose; // this is needed later
private void CloseFile_Click(object sender, System.EventArgs e)
{
        if((currentmenu!=null)
                &&(wpdllInt1.Memo.TextCommand(3,0,0)!=0) // CANCLOSE
           )
        {
           if(WindowMenu.MenuItems.Count>1)
           {
                string s;
                s = currentmenu.RTFID;
                currentmenu.Dispose();                                          TagMenuItem_Click
(WindowMenu.MenuItems[0],null);
                wpdllInt1.Memo.RTFDataDelete(s);
           }
           else
           {
                  wpdllInt1.Memo.Clear(false, false);
                  wpdllInt1.Memo.TextCommand(5,0,0);
            }
        } else AbortClose = true;
}
```

It would be nice to update the menu captions show the current file name when the file was saved under a new name. We also insert an asterix if the file was modified.

```
private void wpdllInt1_OnNotify(object Sender, int Editor, int MsgID)
{
        if ((currentmenu!=null)&&((MsgID==27)||(MsgID==28)))
                currentmenu.Text =
                ((wpdllInt1.Memo.Modified)?"*":"")+
                      wpdllInt1.Memo.LastFileName;
}
```

5.3.6.1 Add Tabset

Now, the icing - add a tabset to switch between documents.

We only need to add the EditorGUI.wpguiPanelH2 flag

```
wpdllInt1.SetEditorMode( ...
                  EditorGUI.wpguiPanelH1
                | EditorGUI.wpguiPanelH2,
                  EditorGUI.wpguiDontSet);
```

and an event handler which handles the click on the tabset buttons (typ=9). It simply takes the name of the button and uses it to select the text.

```
private void wpdllInt1_OnButtonClick(object Sender, int Editor, WPDynamic.IWPDllButton Def)
{
 if(Def.Typ==9)
 {
        string s = Def.Name;
        for(int i=0;i<WindowMenu.MenuItems.Count;i++)
        if(((TagMenuItem)WindowMenu.MenuItems[i]).RTFID.CompareTo(s)==0)
        {
               TagMenuItem_Click(WindowMenu.MenuItems[i],null);
               break;
        }
 }
}
```

5.3.6.2 Hide some buttons

The toolbar shows the new, load and save button. If you want to change the toolbar you can of course use the
application WPImagePack 424 to edit the buttons resource file "buttons.pcc".

But you can also just hide one ore more buttons which listed in the XML layout description.

To do so you can use wpaSetFlags. This method can only be used inside the OnUpdateGUI event.

```
private int wpaNew = -1;
private void wpdllInt1_OnUpdateGUI(object Sender, int Editor, int UpdateFlags, int StateFlags, int
PageNr, int PageCount, int LineNr)
{
       byte[] states = new byte[1];
       if (wpaNew<0) wpaNew = wpdllInt1.wpaGetID("New");
       states[0] = 4; // bit 3 - hide it
       wpdllInt1.wpaSetFlags(Editor,wpaNew,1, states);
}
```

5.3.6.3 Ask to save before close

In the function CloseFile_Click we already let TextDynamic ask the user if the file should be saved or not. This is done by **wpdllInt1.Memo.TextCommand(3,0,0)** - if result is not 0 the user has either saved the file or has chosen not to save.

We can now add an event handler to OnClosing which utilizes the boolean AbortClose we have created before.

```
private void WinForm_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
       AbortClose = false;
       for(int i=WindowMenu.MenuItems.Count-1;!AbortClose&&(i>=0);i--)
       {
```

```
          TagMenuItem_Click(WindowMenu.MenuItems[i],null);
          CloseFile_Click(null,null);
        }
        if(AbortClose)e.Cancel = true;
}
```

Now this message will be triggered for each file:

5.4 Configure the Editor / FAQ

In this topic we list the important methods to configure the editor in TextDynamic.

1) How to select the editor mode, switch between single editor, double editor and split screen?

You need to use the method SetEditorMode 114. Here you specify which mode you need. Since this procedure recreates the editor, please set the license 18 keys before hand.

2) How to select the large or small toolbar?

This is also done by the method SetEditorMode 114, please use the parameter XMode

3) How can I change the rulers to Inches?

This is also done by the method SetEditorMode 114, parameter GUI1 and GUI2 (for first or secondary editor). Use the bit #10 (value 1024). Using the GUI flags you also activate the tool panels and the rulers.

4) How to switch the paragraph symbol " ¶ " off and on?

This flag is part of the "ViewOptions". You can use this method to set different modes:

Memo.SetBProp(6, 29, -1) 201

Please also see the other possible flags for "viewOptions".

5) How to change the toolbar design and color?

Please read here: Change Toolbar Design 107

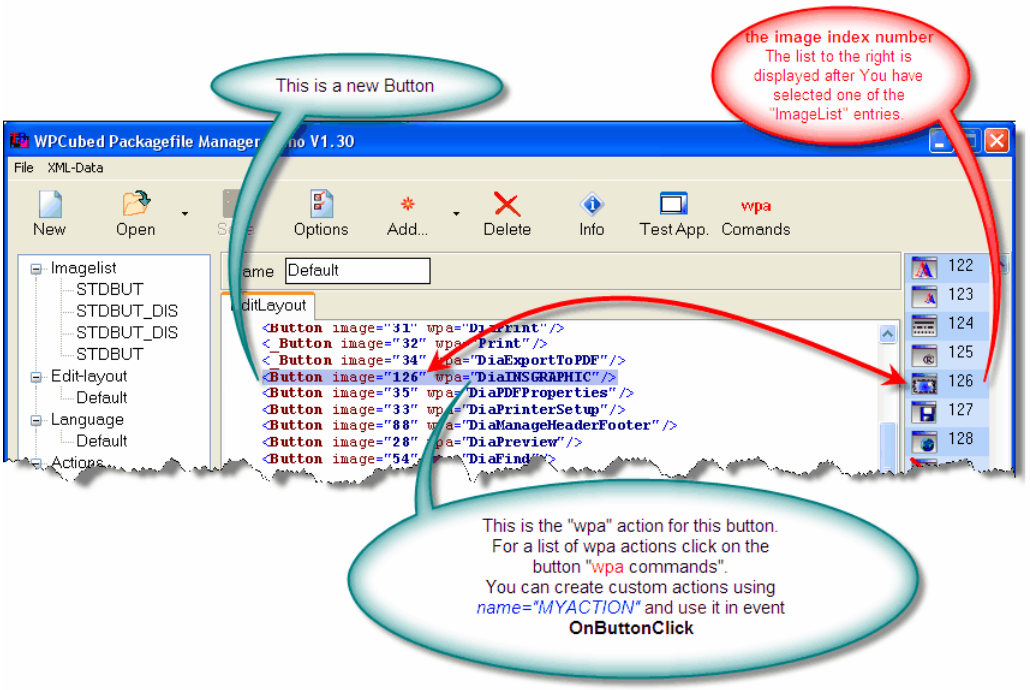
6) How to modify the toolbar?

a) How to add a new button?

The toolbar loads the layout from the "PCC" file - see SetLayout() 57. You can modify the layout using the provided "ImagePack" program. (more... 424)

When you have started the tool WPImagePack.exe you can edit the XML data which describes the toolbar:

Example: We add a new button to show the "insert graphic" dialog.

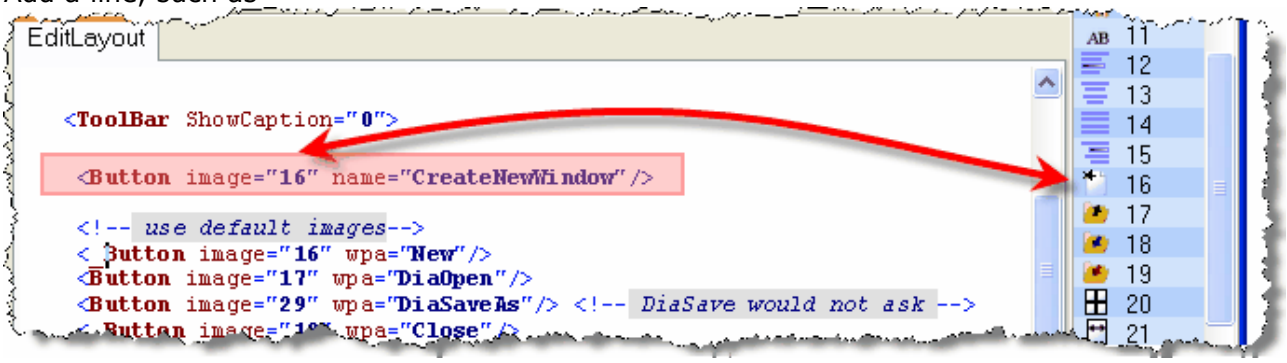


b) How to exchange a glyph (button image)?

It is also possible to replace the glyphs. You need a PNG editor for this. Unlike the demo edition, the registered version can extract the PNG data from the PCC file to be modified externally. (more... 424)

c) How to add a button with custom functionality?

Add a line, such as



then you need an event handler for OnButtonClick 135

```
Private Sub WPDLLInt1_OnButtonClick(ByVal Editor As Long, ByVal Def As WPTDynInt.IWPDllButton 303)
   If Def.Name 305 = "CreateNewWindow" Then
     LoadNewDoc ' implemented as public routine in 'Main.BAS'
   End If
End Sub
```

7) How to activate/deactivate the premium features (text boxes, foot notes and columns) ?

Please use the method SetEditorMode 53.

8) Provide the User with information about cursor position (status bar)

Please use the event OnUpdateGUI. (Example 38)

9) Add a menu to the window and execute actions in the editor

Please use your IDE to create a menu. You can then execute the wpaProcess() to start a certain action. Of course you can also execute the methods in the Memo 160 or TextCursor 219 interface. To automatically enable/disable your menu items use the wpaGetFlags 61 method within OnUpdateGUI 63.

10) Add spell check

The method Memo.TextCommand(7, ParamA, 0) 208 can be used to control the spell check. Please also see the spell check category 160 and the VB6 example in the topic IWPSpell 388.

11) Add hot keys / shortcuts

You can add short cuts using the KeyPress 144 event. Please see example there.

12) Disable editing

You can set the property Memo.Readonly 174 to true. Also see the options available via Memo.SetBProp 197

13) How to resize the editor automatically according to the size of the window

This mode is called "WradWrap Mode". You can activate it using Memo.WordWrap 175.

6 Introduction to the API

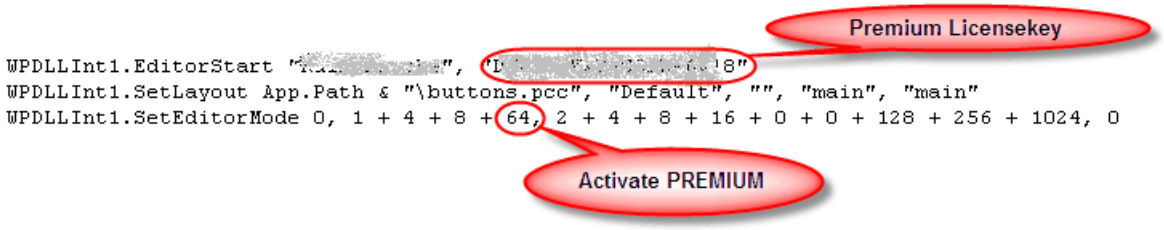
6.1 SetEditorMode()

TextDynamic can work in several different ways. The mode is selected by using the method WPDLLInt1.SetEditorMode(). **One Control can host two editors at once** - we know no other word processing control which does this, so why does TextDynamic?

Sometimes you want to see two areas of the same text at the same time. For example the start and the end of a longer text. You can easily jump between those areas and edit both. This is called SplitScreen. For this mode you need two editors which edit the same text.

Sometimes you also need to work with texts which share properties. In our control such a case is the report template and the result - both share the same paragraph styles. Or mailmerge - the original form and the long text which consists of multiple appended letters. Also in this case it is a big advantage if the text styles are shared. If the user changes the text style in the source editor the destination editor will be also be updated.

Note: SetEditMode is used to activate the "premium" and "spellcheck" features - if included in your license:

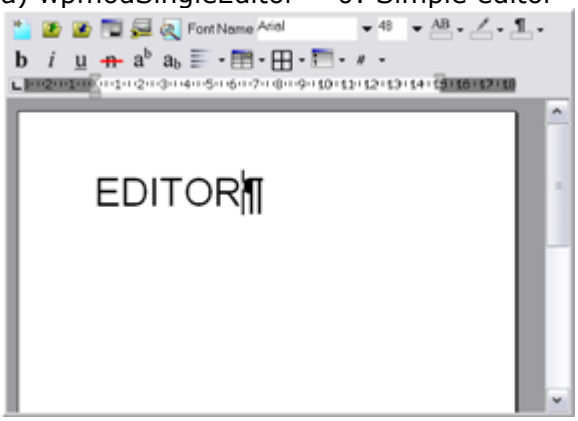


The method SetEditorMode 114() requires 4 parameters.

Examples:
SetEditorMode - Example C# 56
SetEditorMode -Example VB6 57

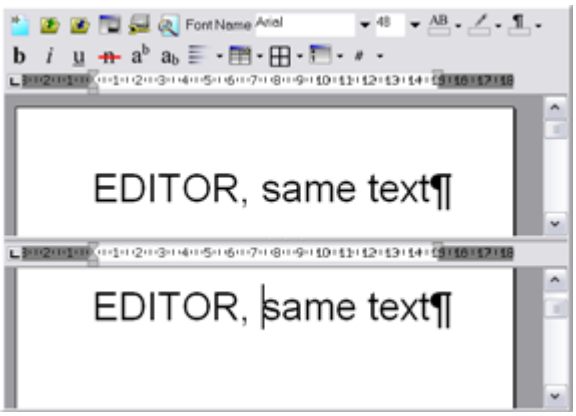
1. Parameter: "Mode" (.NET: enum EditorMode)

a) wpmodSingleEditor = 0: Simple editor

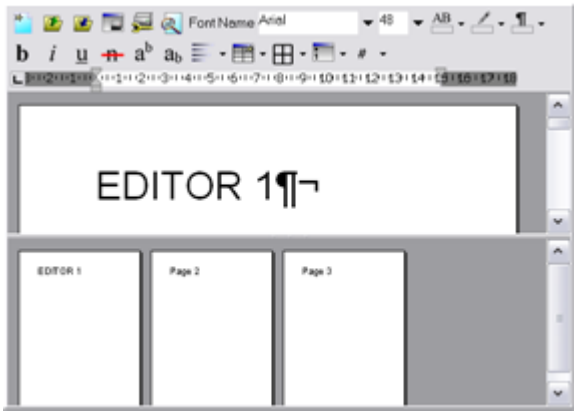


c) wpmodDoubleEditor = 1: two Editors - working with 2 different texts. Both text are using the same paragraph styles. This mode is required if you use the TextDynamic reporting feature 87. In one editor the report template would be displayed, the other displays the created report. Usually the DLL works with the upper, Editor 1. To switch to Editor 2 the method SelectEditor can be used. Most of the events pass a parameter 'editor' which is 1 for the upper, 2 for the lower editor.

b) wpmodSplitEditor = 2: Editor with split screen - the same text is edited in both windows. But each window can use a different zoom level and layout mode. Both windows can show a different position in the text.Most of the events pass a parameter 'editor' which is 1 for the upper, 2 for the lower editor.



d) wpmodSplitThumbnails = 3: Editor and Preview/Thumbnail display. This mode is almost identical to the 'split screen' mode. The lower editor is readonly though.



2. Parameter: "ModeX" (.NET: enum: EditorXMode)

In addition to the general operation modes certain special features can be activated using property ModeX.

This is a "bit field". Certains bits are set for certain functionality. When using VB you have to take the number values and add them. In C# you use the or "|" operator to combine the bits.

The following flags are possible:

wpxtraToolbar Select the 16x16 toolbar
wpxtraToolbarLG = 2 - Select 24x24 Toolbar
wpxtraSpellcheck = 4 - Activate the Spellcheck
wpxtraTables = 8 - Activate support for tables (in toolbar)
wpxtraPDFExport = 16 - Activate the wPDF PDF Export
wpxtraReporting = 32 - Activate the WPReporter in EditorA (cannot be used in 'Split' mode!)
wpxtraPremium = 64 - Activate the premium features, text box and foot note support

Please note that setting a flag will have no effect if you do not have the license to use i.e. the PDF export.

3. Parameter: "Gui1" (.NET: enum EditorGUI)
This parameter selects the **g**raphical **u**ser **i**nterface elements used by the editor 1.

```
wpguiRuler       = 2, // Select a ruler
wpguiVertScrollBar=4, // Select the vertical scrollbar
wpguiHorzScrollBar=8, // Select the horizontal Scrollbar
wpguiVertRuler   =16, // Select Vertical Ruler
```

```
wpguiGutter     =32, // Select Gutter (pagno)
wpguiPanelV1    =64, // Select the panel in top right corner
wpguiPanelV2    =128,// NAVIGATION: Select the panel in the bottom rigth corner
wpguiPanelH1    =256,// VIEW: Select the panel in the bottom left corner
```

If the flag **wpguiDontSet** (1) was used this parameter will be ignored. This is useful if you call SetEditorMode later to modify an editor.

As this is also a bit field please combine the values using **"+"** or **"or".**

4. Parameter : "Gui2"

This parameter selects the GUI elements of the second, the lower editor. Of course this makes only sense if a second editor/viewer was selected using "Mode".

Example - C#:

```
WPDLLInt1.SetEditorMode(
        EditorMode.wpmodSingleEditor,
        EditorXMode.wpmodexToolbarLG,
        // EditorGUI.wpguiGutter +
        EditorGUI.wpguiHorzScrollBar |
        EditorGUI.wpguiPanelH1|
        // EditorGUI.pguiPanelH2 |
        EditorGUI.wpguiPanelV1|
        EditorGUI.wpguiPanelV2 |
        EditorGUI.wpguiRuler  |
        EditorGUI.wpguiVertRuler  |
        EditorGUI.wpguiVertScrollBar
        , EditorGUI.wpguiDontSet);
```

6.1.1 SetEditorMode - Example C#

Example C#

This image shows initialization code written in C# (VS 2005). We had simply dropped a WPDLLInt control on the form and changed its name to "WPDLLInt1"

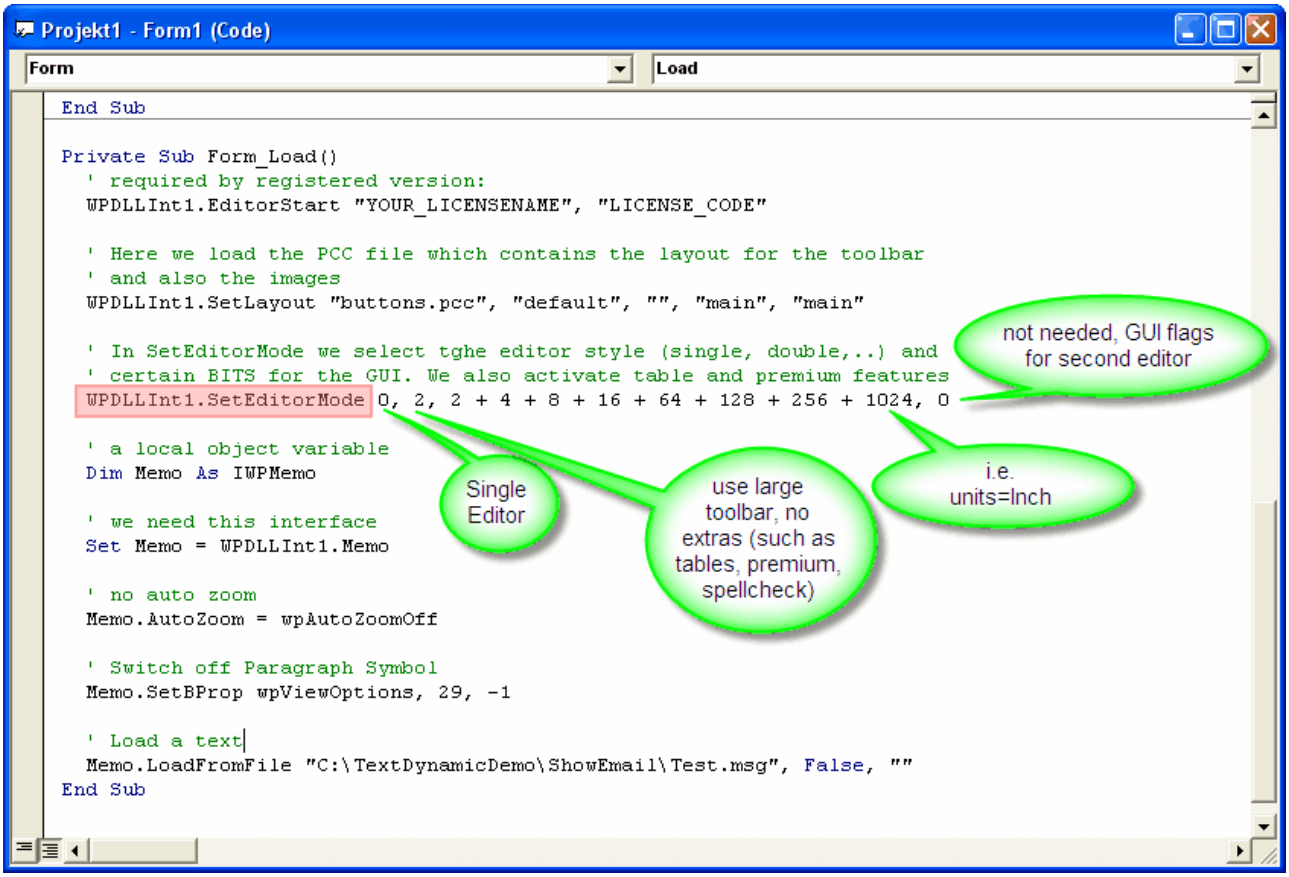
```
using System.Text;
using System.Windows.Forms;
using WPDynamic;  // reference to the TextDynamic wrapper

namespace WindowsApplication1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
          WPDLLInt.SetDLLName("{hkcu}Software\\WPCubed\\TextDynamic\\path");

         InitializeComponent();
```

```
// 1.) Set the Editor Mode
WPDLLInt1.SetEditorMode
(
  EditorMode.wpmodDoubleEditor,
  EditorXMode.wpmodexToolbarLG,  // = ModeX
  EditorGUI.wpguiHorzScrollBar | // = GUI of upper Editor
  EditorGUI.wpguiPanelH1 | EditorGUI.wpguiPanelV1 |
  EditorGUI.wpguiPanelV2 | EditorGUI.wpguiRuler |
  EditorGUI.wpguiVertRuler | EditorGUI.wpguiVertScrollBar
  , EditorGUI.wpguiVertScrollBar  // = GUI of lower Editor
);
// 2.) Load the PCC file
WPDLLInt1.SetLayout("..\\buttons.pcc", "");
```

6.1.2 SetEditorMode -Example VB6

Example VB6:

When you develop in VisualBasic 6 a good place to do the initialization is the event Form.OnLoad.



6.2 SetLayout()

The method SetLayout() is used to load the description file for the user interface. The editor cannot display a toolbar without such a file. The description file is created by the "Packagefile Editor 424".

SetLayout() expects 5 parameters:

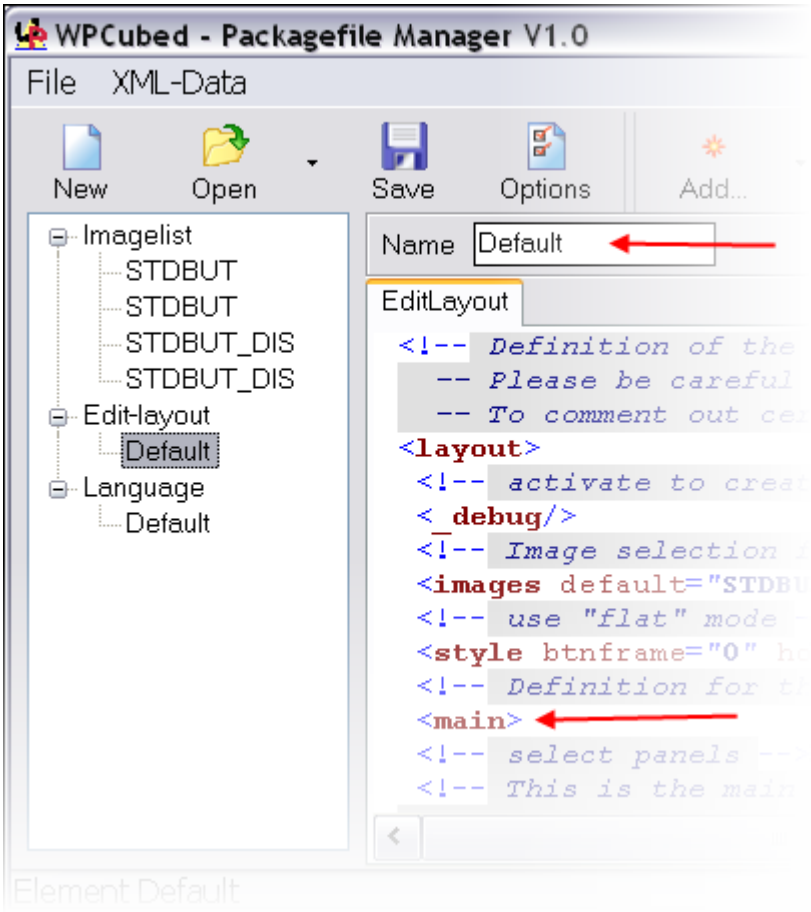
```
1. string LayoutFile - the path to the layout file, i.e. "Buttons.PCC"
2. string LayoutName - the name of the layout entry in the package file, i.e.
"default"
3. string LayoutPW - the password used for the PCC file or an empty string
```

```
4. string LayoutMainXML - the XML branch for the main editor, i.e. "main"
5. string LayoutSecondXML - the XML branch for the second editor (can be the
same as LayoutMainXML)
```

The .NET assembly also implements the SetLayout method with only two parameters, LayoutFile and LayoutPW. The other parameters will use the values "default", "main", "main".

Example:

For a package file which uses this names ...



... the call to SetLayout has to look like this:

```
C#:   SetLayout(116)("..\\buttons.pcc", "default", "", "main", "main");
```

or, shorter:

```
      SetLayout("..\\buttons.pcc","");
```

```
VB6:

Private Sub Form_Load()
   ' The PCC file will be searched in the same path as the DLL !
   WPDLLInt1.SetLayout "..\buttons.pcc", "Default", "", "main", "main"
End Sub
```

Please note: SetLayout is not required if you do not use the toolbar or any of the tool panels. In this case no PCC file needs to be loaded.

If you specify a relative path, the directory which contains the engine DLL will be used as basis.

6.3 Update design of toolbars

TextDynamic supports different "themes" for the dialogs, toolbar and the panels.

Please also see the overview in "Configure the Editor 51".

They can be globally updated using the **Command** method.

This command ids are defined:

WPDLL_COM_DialogBackground = 9501

Activate a brushed metal background for the dialogs.

"Standard" 

"Metal" 

You can also load a background image for the tiling

```
wpdllInt1.Command(9501,1,"c:\\someimage.bmp");
// You can use the token {dll} in the filename
// to specify the path relatively to the engine DLL.
```

WPDLL_COM_ToolbarDesign = 9502

Select the design style for all toolbars which do not use the <design> tag in the XML.

This modes are supported:

1 = Titanium (default)

2 = Brushed Metal - use Command(9501, 1) to make the dialogs use this background

3 = Shaded. Use Command 9503 and 9504 to change colors)

4 = Simple gray

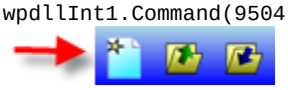
WPDLL_COM_ToolBarColorFrom = 9503

Change the start color for the gradient fill used by mode 3. Default is $FFFFFF

WPDLL_COM_ToolBarColorTo = 9504

Change the end color for the gradient fill used by mode 3. Default is $FE9696

```
C# Example:
wpdllInt1.Command(9502,3,0);
wpdllInt1.Command(9503, wpdllInt1.ToRGB(Color.LightBlue),0);
wpdllInt1.Command(9504, wpdllInt1.ToRGB(Color.Blue),0);
```



WPDLL_COM_SetHorzBack = 9505

Load a bitmap which is stretched on the background of horizontal toolbars. You can realize individual shading using such a bitmap.

WPDLL_COM_SetVertBack = 9506

Load a bitmap which is stretched on the background of vertical toolbars

6.4 SetEnableFlags

Using the method SetEnableFlags it is possible to modify the editor #1 or #2:

Certain options which are not active by default can be activated by setting one bit in the parameter:

bit 1: wpoptDropImgCreatesLinkedImage=1 - when an image file is dragged onto the editor a link to the file is created. (only the path name is stored).

bit 2: wpoptDropImgCreatesMovableParObject=2 - when an image file is dragged onto the editor a movable image will be created - it is positioned relatively to a paragraph.

bit 3: wpoptDropImgCreatesMovablePageObject=4 - when an image file is dragged onto the editor a movable image will be created - it is positioned relatively to a page.

bit 4: wpoptDropImgCreatesNoWrapImage=8 - when an image file is dragged onto the editor a movable image will be created with text wrapping switched off.

bit 5: wpoptFormularMode=16 - Activates the formular mode. The user can only edit certain mail merge fields which are marked to be editable (Mode=2).

bit 6: wpoptAllowCreateTableInTable=32 - Tables may be created within other tables.

bit 7: wpoptShowSpecialChars=64 - Special characters, such as CR, NL and FF are displayed.

Please also see method **SetBProp** which allows it to toggle more than 100 internal proerty flags to adjust the editor handling to you needs.

6.5 SetDisableFlags

Using the method SetEnableFlags it is possible to modify the editor #1 or #2:

Certain options which are active by default can be **de**activated:

```
wpoptNoEdit          =     1;   // editor is readonly
wpoptNoSelect        =     2;   // selection not visible
wpoptNoFileDrop      =     4;   // Don't accept image files from Explorer
wpoptNoDragDrop      =     8;   // Disable Drag&Drop
wpoptNoCopy          =    16;   // Disable Copy
wpoptNoPaste         =    32;   // Don't allow any paste
wpoptNoPasteIMG      =    64;   // Don't allow paste of images
wpoptNoPasteRTF      =   128;   // Don't allow pase of RTF (or HTML)
wpoptNoEditHeader=   512;   // Don't switch to header/footer to edit
wpoptNoResizeImages    = 1024; // User cannot resize images
wpoptNoResizeTables    = 2048; // User cannot move or resize tables
wpoptNoResizeColumns   = 4096; // User cannot resize columns
wpoptNoResizeRows      = 8192; // User cannot resize Rows
```

Note: The interface IWPMemo also includes the method SetBProp - this method allows it to set docends of other flags. Please see online reference.

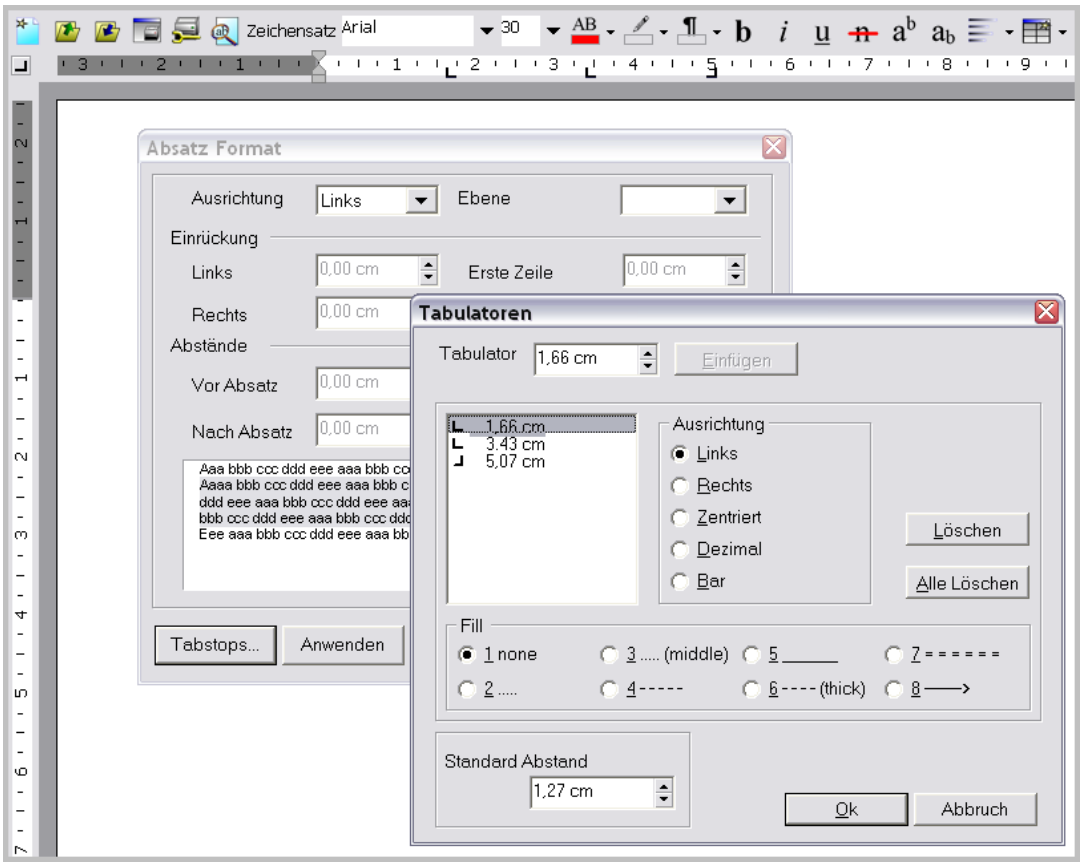
6.6 SetLanguage (Localization)

This method selects the language for the user interface.

The strings used by the user interface are defined in a special XML section in the package file which also contains the description of the toolbar and the toolbar images. (see SetLayout 57 and LanguageFile 438)

Example: WPDLLInt1.SetLanguage("DE") will select German language.

This are the dialogs to change the paragraph indents, tabs and spacing - with German language selected.



6.7 wpaGetFlags

The method wpaGetFlags returns the state of all available "wpa 419" actions in one array.

This is especially useful if you use pre calculated wpa action ids. You can for example on startup assign the id to each menu/button you use and later use the stored id to execute the action (wpaExec)

In the array the bits of each element represent:
bit 1: action is enabled
bit 2: action is selected (menu shows check, button is pressed)
bit 3: action is hidden, it is not available.
bit 7: This bit is always set to avoid #0 entries

VB6:
The method wpaGetFlags of TextDynamic OCX returns a string

```
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
s = WPDLLInt1.wpaGetFlags(0) ' this are the flags for the current editor
i = WPDLLInt1.wpaGetID("bold") ' this is the id for the command to toggle 'bold'
Bytes = StrConv(s, vbFromUnicode) ' convert string to bytes

If Bytes(i) And 2 Then BoldMenu.Checked = True Else  BoldMenu.Checked = False
```

C#:
The method wpaGetFlags of TextDynamic OCX returns a byte array

This example event handler for the event OnUpdateGUI 63 modifies the button responsible to switch italic on and off.

```
private void wpdllInt1_OnUpdateGUI(object Sender,
```

```
   int Editor, int UpdateFlags, int StateFlags, int PageNr, int PageCount, int LineNr)
{
        byte[] stateflags = wpdllInt1.wpaGetFlags(0); // Current editor
        btItalic.Pushed = (stateflags[wpdllInt1.wpaGetID("Italic")] & 2)!=0;
}
```

This just a simplified example. We recommend to use wpaGetID once at startup to get the IDs for all used actions and use them later, not only for wpaExec (to execute an action) but also as index in the array created by wpaGetFlags.

Using this logic you can update of your user interface - without wasting resources by too frequent updates or exchanging data with the editor code too often.

6.8 wpaSetFlags

The method wpaSetFlags can be only used within the OnUpdateGUI event.

With it you can manipulate the states of the buttons in the editor i.e. hide or disable some.

In the array the bits of each element represent:
bit 1: action is enabled
bit 2: action is selected (menu shows check, button is pressed)
bit 3: action is hidden, it is not available.
bit 7: This bit is ignored, you can set it to avoid #0 entries

You can use wpaGetFlags to first get the states of all buttons and the use wpaSetFlags to change the flags, or you use a new bytes array (with OCX use a string) only for the action you need to update.

Note: If the "start" parameter is >0, the first element of the passed array will be mapped to the flag property of the action with the id "start".

C#:

In this example we hide the button "new" - (we first use wpaGetFlags).

```
private int wpaNew = -1; // we store the result value of wpaGetID one time
private void wpdllInt1_OnUpdateGUI(object Sender, int Editor, int UpdateFlags, int StateFlags, int
PageNr, int PageCount, int LineNr)
{
        byte[] states;
        states = wpdllInt1.wpaGetFlags(Editor);
        if (wpaNew<0) wpaNew = wpdllInt1.wpaGetID("New");
        states[wpaNew] |= 4; // bit 3 - hide it
        wpdllInt1.wpaSetFlags(Editor,0,states.Length, states);
}
```

Since we only need to hide one button we can also write

```
private int wpaNew = -1;
private void wpdllInt1_OnUpdateGUI(object Sender, int Editor,
               int UpdateFlags, int StateFlags, int PageNr, int PageCount, int LineNr)
{
        byte[] states = new byte[1];
        if (wpaNew<0) wpaNew = wpdllInt1.wpaGetID("New");
        states[0] = 4; // bit 3 - hide it
        wpdllInt1.wpaSetFlags(Editor,wpaNew,1, states);
}
```

VB6:

When using the OCX the array is passed as standard unicode string. Since bit 7 can be set, we can write the "hide" state as "E". The standard "enabled" state equals "A", disabled is "@".

Above example written in VB6

```
Option Explicit
Dim wapNew As Integer

Private Sub WPDLLInt1_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)
  If wapNew = 0 Then
     wapNew = WPDLLInt1.wpaGetID("New")
  End If
  WPDLLInt1.wpaSetFlags 1, wapNew, 1, "E"
End Sub
```

6.9 Event OnUpdateGUI

The event OnUpdateGUI will only be called in idle phases to give the application a chance to

a) update the status bar information,
b) change the Pushed state of buttons
c) enable or disable menu items.

Es mentioned before it is used best in combination with wpaGetFlags.

int It receives 6 parameters:

Editor : This is the current editor, 1 or 2

int UpdateFlags : This is a bit field.

value 1: Selection has been changed
value 2: Undo State has been changed
value 4: The Paragraph Attributes have been changed
value 8: The Character attributes have been changed
value 32: The Cursor piosition has been changed

int StateFlags, also a bit field

value 1: the modified flag for the text
value 2: the editor is in inserting mode
value 4: UNDO is possible
value 8: REDO is possible
value 16: Text is selected
value 32: Currently an object is selected
value 64: The selected object is an Image
value 128: Currently the cursor is not defined (Should not happen)
value 256: The Cursor is inside of a table
value 512: The current paragraph uses a style sheet
value 1024: The cursor is in header or footer texts (not in text body)

int PageNr

The page number the cursor is located within

int PageCount

The current page count

int LineNr

The line number (on the current page)

6.10 Properties

6.10.1 string Text

This property reads and writes the text inside the main editor (1). Using the property TextFormat it is format to set the format for the text, i.e. "RTF" (default), "HTML", "ANSI" or "UNICODE".

For long texts we recomend to use files or stream methods to load and save. This methods are available in the IWPMemo 163 interface, see property "Memo", "Memo2" and "CurrMemo".

6.10.2 string Text2

This property reads and writes the text inside the lower editor (2). Using the property TextFormat it is format to set the format for the text, i.e. "RTF" (default), "HTML", "ANSI" or "UNICODE".

6.10.3 string TextFormat

This property selects the load and save format used by the properties Text and Text2. You can select the type ("RTF") and also append certain options, such as "RTF-onlybody" if you want to save the RTF text but no header or footer. Also see the list of format strings 150.

6.10.4 IWPPdfCreator PDFCreator

This property provides a reference to the PDFCreator 271 interface. This interface can be used to initialize and start the PDF creation. The reference will ne null in case your license (or the DLL) does not contain PDF creation.

This code will export the text in Editor #1 to PDF. PrintSecond() will export from editor 2.

```
IWPPdfCreator PDFCreator = WPDLLInt1.PDFCreator;
if (PDFCreator!=null)
{
   PDFCreator.PDFFile = "c:\\pdffile.pdf";
   PDFCreator.Print();
} else MessageBox.Show("PDF Creation not available");
```

If you need to create multiple pages through mail merge use code such as this C# example.

```
IWPPdfCreator PDFCreator = WPDLLInt1.PDFCreator;
if (PDFCreator!=null)
{
        WPDLLInt1.PDFCreator.PDFFile = "c:\\testa.pdf";
        WPDLLInt1.PDFCreator.BeginDoc();
        GotoFirstDataRow();
        while (! AtEndOfData() )
        {
                WPDLLInt1.Memo.MergeText("");
                WPDLLInt1.Memo.ReformatAll(false,false);
                WPDLLInt1.PDFCreator.Print();
                GotoNextDataRow();
        }
        WPDLLInt1.PDFCreator.EndDoc();
} else MessageBox.Show("PDF Creation not available");
```

The call to ReformatAll is required to reformat the text inside this loop. Usally the text is not formatted until the Application is idle. If you export right after a file was loaded, ReformatAll also must be used. The functions printed in green are placeholders.

6.10.5 IWPMemo Memo

This is probably the most important property. "Memo" provides an interface to the editing interface of the upper editor #1 and "Memo2" provides the interface to program the lower editor. "CurrMemo" can be used to access the editor was focussed last.

Note: Our product RTF2PDF V3.5 or later publishes also the properties "Memo" and "Memo2". Here the interface IWPEditor is used which contains most of the IWPMemo methods. So code written for TextDynamic can be converted to use RTF2PDF easily.

The interface IWPMemo 160 itself provides access to other interfaces, numerous methods and properties.

IWPMemo Memo2

The interface is identical to "Memo" but it works with the optional editor #2.

IWPMemo CurrMemo

This is the 'memo' interface of the current (selected) editor.

IWPTextCursor Memo.TextCursor

The IWPMemo also includes a reference to IWPTextCursor 219.

This interface is used to position the cursor and to create text under program control.

IWPAttrInterface.TextAttr

This interface 276 modifies the current writing mode, for example selects the font face. If the user has selected some text, any change will modify the selected text.

IWPPageSize PageSize

This interface 333 allows the modification of the page size. Please note, using the event OnMeasurePage 131 it is possible to have different page sizes for each page in the document.

6.10.6 IWPCharacterAttr SpecialTextAttr()

These interfaces make it possible to modify the look of "special text".

This special text can by hyperlinks, fields, bookmarked text etc.

To display all text objects with a yellow background this code can be used:

```
WPDLLInt1.SpecialTextAttr(
      SpecialTextSel.wpFieldTextObjects).BackgroundColor = WPDLLInt1.ToRGB(Color.Yellow);
```

This code displays the hyperlinks underlined with a red background:

```
WPDLLInt1.SpecialTextAttr(SpecialTextSel.wpHyperlink).Underline =  ThreeState.tsFalse;
WPDLLInt1.SpecialTextAttr(SpecialTextSel.wpHyperlink).BackgroundColor = WPDLLInt1.ToRGB(Color.Red);

public enum SpecialTextSel
{
        wpHiddenText, // text which is hidden (RTF \v)
        wpFootnote, // footnote numbers
        wpInsertpoints, // start and end markers of mail merge fields
        wpHyperlink, // hyperlinks
        wpSPANStyle, // start/end SPAN objects (used by HTML)
        wpAutomaticText, // the merged text itself (insife merge fields)
        wpProtectedText, // protected text
        wpBookmarkedText, // text within bookmark tags
        wpInsertedText, // not used
        wpDeletedText, // not used
```

```
        wpWordHighlight, // not used
        wpFieldTextObjects // field objects, such as a page number
}
```

6.10.7 IWPAttrInterface AttrHelper

This interface can be used to calculate CharAttr index values. Such a value is an index in a certain character attribute cache which contains records with font attributes for text. Please note that 'Clear' invalidates all character attribute index values.

You can use the CharAttrIndex with IWPParInterface 339.SetText or IWPTextCursor 219.InputString.

6.10.8 int SelectedEditor

Using this property the currently active editor can be changed. The following properties will access the respective active editor.

6.10.9 IWPTextCursor TextCursor

The cursor interface 219 is used to create text under program control and to position the cursor. This interface provided by this property is the same as 'CurrMemo.TextCursor' 168.

Create some text and display it:

```
IWPTextCursor TextCursor;
TextCursor = WPDLLInt1.Memo.TextCursor;
TextCursor.InputText 251("Hello World\nand a new line ....");
WPDLLInt1.Memo.Reformat();
```

Insert a Picture:

If you want to insert a picture which is currently displayed in a .NET PictureBox you need to use the class WPDynamic.Image2Picture.

```
WPDLLInt1.Memo.TextCursor.InputPicture(
        new WPDynamic.Image2Picture(pictureBox1.Image),0,0);
```

This are the methods to insert images. The result value is 0 if no image was insered:

a) int InputImage 241(string filename, int Mode);

The image is loaded from a file. The following formats are supported: BMP, WMF, EMF, JPEG, PNG.

If bit 1 was set in parameter 'mode' the image will not be embedded, only a link to the file will be stored.

b) int InputPicture 244(IUnknown Picture, int Width, int Height);

This method inserts a picture defined by either a IPicture interface (common in VB6) or, under .NET, the interface IWPPicture which is provided by the utility class Image2Picture. IWPPicture does, unlike IPicture, also handle PNG and JPEG data.

c) int InputPictureStream 244(IUnknown Stream, string FileExt, int Width, int Height);

This method inserts image data stored in a stream. The property FileExt tells the method the format of the data for example "BMP".

The parameter Stream can be either an IStream or IWPStream interface. The latter is published by the helper class **Stream2WPStream** which is implemented in the C# wrapper.

In this example we 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.

Insert a Table

To create a table there are four options.

a) use InputTable 249, InputRowStart, [InputCell....] InputRowEnd
b) use AddTable 224 and the event OnCreateNewCell which is called for each created cell
c) use AddTable 224 to create the first row and then append to that table using CPMoveNextRow 229(true)
d) use low level ParInterface 339 methods.

Please see the table support category 158.

Which option is better depends on the problem you need to solve. Option requires less coding. But it requires more abstraction and also that the data which should be inserted is available in event handler as well. The option a) is good if you do not know in advance the count of rows you need to create. It also makes it possible to create tables with a different column count in each row. The option c) mimics the way somebody would create a table "by hand". d) is a versatile approach which directly works with the paragraph objects. It allows the creation of a table without moving the cursor position.

6.10.10 IWPAttrInterface CurrAttr

This interface allows the modification of the current writing mode. It changes the current writing mode. If you need to change the selected text or, if no text is selected, the current writing mode - this is what tool bar buttons usually do - use property TextAttr 68.

*The interface **IWPAttrInterface** has several methods to read and write properties, i.e. AttrGet and AttrSet. These methods expect the property ID (WPAT...) - the manual (PDF) contains a 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 C# example code creates text to show the possible underline modes.

```
string[] cnames =  new string[]
{ "WPUND_Standard"
, "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();
```

6.10.11 IWPAttrInterface TextAttr

If you need to change the selected text or, if no text is selected, the current writing mode use property TextAttr 162. You can use it to create your own toolbar, 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.

6.10.12 IWPAttrInterface CurrSelAttr

If text is selected, this interface 165 makes it possible to modify the attributes of the selected text.

6.10.13 IWPParInterface CurrPar

This interface 164 allows modification of the current paragraph, this is the paragraph the cursor is located in.

If you use the method TextCursor.InputTable, StartTableRow, InputCell, EndTableRow this property allows it to change the properties of the current paragraph/cell, table or table row.

Please note, that, even if you store the reference of the CurrPar interface to a variable, the interface will always modify the paragraph which is 'current' at the time you access methods of the interface. The same is true for the interfaces CurrParAttr.

The IWPMemo interface also publishes the interfaces CurrPar, CurrParAttr etc.

This interfaces are also published by IWPRTFDataBlock 296 - this makes it possible to modify the text without moving the cursor!

This C# example appends formatted text:

```
IWPParInterface par = wpdllInt1.memo.CurrPar;
IWPAttrInterface atr = wpdllInt1.AttrHelper;

// Append normal text
   atr.Clear();
   par.AppendText("Normal ", atr.CharAttrIndex);
// bold text
   atr.IncludeStyles(1);
   par.AppendText("and bold", atr.CharAttrIndex);
```

This C# example adds arabic numbering to the current paragraph

```
IWPParInterface par;
par = WPDLLInt1.CurrPar;
if(par!=null)
{
      par.IndentLeft  = (int)(0.5 * 1440); // 1/2 inch
      par.IndentFirst = -(int)(0.5 * 1440);
      par.NumberMode  = 3;
}
```

You can modify each of the propeties which is defined by a WPAT-code 410 using the method ParASet and delete it using ParADel. To modify the attributes of the text use CurrParAttr or, for each individual character, CurrPar.CharAttr(index).

This code creates a table using TextCursor.InputTable. This is just one possibility to create a table. The other high level method is TextCursor.AddTable which requires a callback to set properties and fill in text. InputTable does not require a callback which makes it easy to use in interpreted code.

```
      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);
```

6.10.14 IWPAttrInterface CurrParAttr

This interface allows modification of the text attributes of the current paragraph, the paragraph the cursor is located in. **Any change will update all characters in this paragraph!** It will not add or remove the properties on the paragraph object level, but modify the character attributes of each character in this the text. To update the attributes hosted by the paragraph object use the method ParASet of the property CurrParAttr 164.

You can also access each character individually, use ParrAttr.CharAttr(index).

6.10.15 IWPParInterface CurrStyle

This interface 165 allows modification of the current style, this is the style which was just added or selected 196 .

6.10.16 IWPAttrInterface CurrStyleAttr

This interface is used to modify the character attributes of the current style.

6.10.17 IWPPageSize PageSize

This is the page size the selected editor is using. Using the interface IWPPageSize 333 it is possible to change the page size and the margins.

6.10.18 IWPTextObj CurrObj

This is the interface 396 to the object interface of the current object.

The current object is the object which has been inserted last.

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);
```

6.10.19 IWPTextObj CurrSelObj

This is a reference to the interface to the object which is currently selected. Please note that this reference can be null. As with all the other properties listed above, this will use the editor which is currently active.

Example:

```
IWPTextObj 396 selobj =  wpdllInt1.CurrSellObj;
if ((selobj!=null)&&
          (selobj.ObjType==TextObjTypes.wpobjImage))
{
          selobj.LoadFromFile("c:\\Test.bmp");
}
```

6.10.20 IWPMapi

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 319 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 formated text, first assign the string "<html>" to the property Body .

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 to the e-mail client. (Internally the Windows API MAPISendMail is used. The parameter lhSession is passed as 0, ulUIParam can be set using the method SetAppHandle, lpMessage will be automatically prepared and flFlags can be set using the property MAPIFlags)

Send2: Send the e-mail which was prepared last. If no e-mail was prepared the document from editor 2 will be send to the e-mail client.

Clear: Resets the properties Subject, FromName, FromAddress, Recipients, CCList,BCCList, AttachmentList, Body.

7 Tasks

7.1 Categories in the programmers reference

Character Attributes 146	Methods to read and write the character attributes
Character Styles 147	Methods to read and write the attributes, bold, italic, underlined ...
Paragraphstyle Support 157	Interfaces, Properties and Methods to use Paragraphstyles
Load and Save 150	Methods to load and save the text as file, stream or string.
Hyperlinks and Bookmarks 149	Create and use Hyperlinks and Bookmarks
Image Support 149	Methods to insert and manipulate images
Hader and Footer Support 148	Create and manage header and footer texts
Callback Functions 148	Enumerate all paragraphs, styles or header/footer texts
Table Support 158	Create and modify tables
Mailmerge 154	Update and read out data fields in the text
TextDynamic CSS strings 159	Use 'WPCSS' strings to store and apply attributes
Document Properties 147	Save and Load string variables with documents
Attribute IDs 410	Methods with work with attributes referenced by IDs
Position Markers 154	Methods to temporarily save cursor position
Lowlevel Paragraph IDs 155	Paragraph IDs can be used to change "current" paragraph
Logical MDI Support 153	Manage multiple documents in one control.
Action Names 417	Predefined actions which are used by the toolbar
Display Status Information 157	Show current page, page count, zooming
Coordinate Conversion 147	Methods to retrieve and convert coordinates used inside the editor
Standard Editing Commands 156	Commands usually found in menu "Edit"
Modify the layout of the text display 153	Properties and Methods which change the way the text is displayed
Printing 156	Properties and methods for printing
Reporting 94	With the TextDynamic report a text file (RTF or WPT format) is created which allows full editing, including changing of the page size.

7.2 Load & Save

The following methods show a dialog box to enter a filename:

```
bool Load(string InitPath, string Filter);
bool Save(string InitPath, string Filter);
bool SaveAs(string InitPath, string Filter);
bool LoadEx(string InitPath, string Filter);
```

InitPath is the initial directory fro the dialog. Filter the filter string, i.e. "Textfiles(*.TXT)|*.TXT".

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" and "UNICODE". When reading text the format is detected automatically if an empty string or "AUTO" was specified.

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 proviced 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 225(10) to trigger the PropChanged before any code which updates the text.

7.3 Mailmerge

The mail merge process offered by TextDynamic is one of its most popular features.

To use the mail merge feature you only need to

a) insert fields
Method: TextCursor.InputField 237
b) create callback which fills in data
Event: OnFieldGetText 73
c) start the merge process
Method: Memo.MergeText 189

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 75 and label printing 76 demo which show how to use this powerful feature.

Please see the example for InputCell 235 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)

7.3.1 Create Merge Fields

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);
```

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 233(StartText, EndText,FieldPreText) (=ReplaceTokens) to convert the text into merge fields.

Please also see the method "MergeText 189" and "InputField 237".

Please also see our label printing 76 demo.

7.3.2 Insert Data

To fill a field please use the event OnFieldGetText 129. This event is triggered by the method MergeText 189.

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** 307 has many options, but usually you will only need to modify the property StringValue 309:

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 129 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

7.3.2.1 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
```

7.3.2.2 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")

7.3.3 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;
```

To make the merged text use a gray background use this code

```
WPDLLInt1.SpecialTextAttr(wpAutomaticText).BackgroundColor = &HE0E0E0 ' gray
```

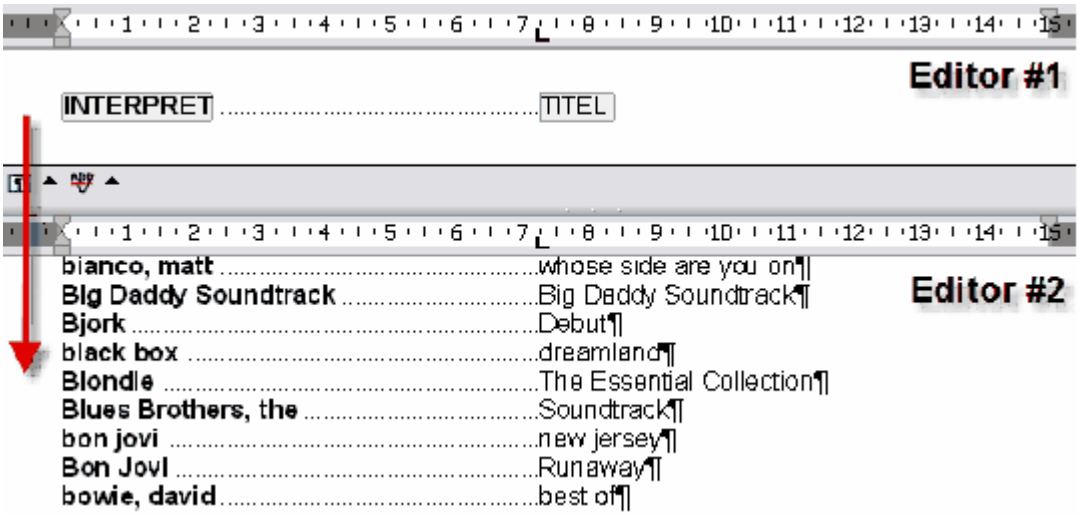
Memo.ShowFields 174 = true

In this mode the field names are displayed in boxes and the merged text is hidden: INTERPRET

7.3.4 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 53 to initialize the "double editor" mode.



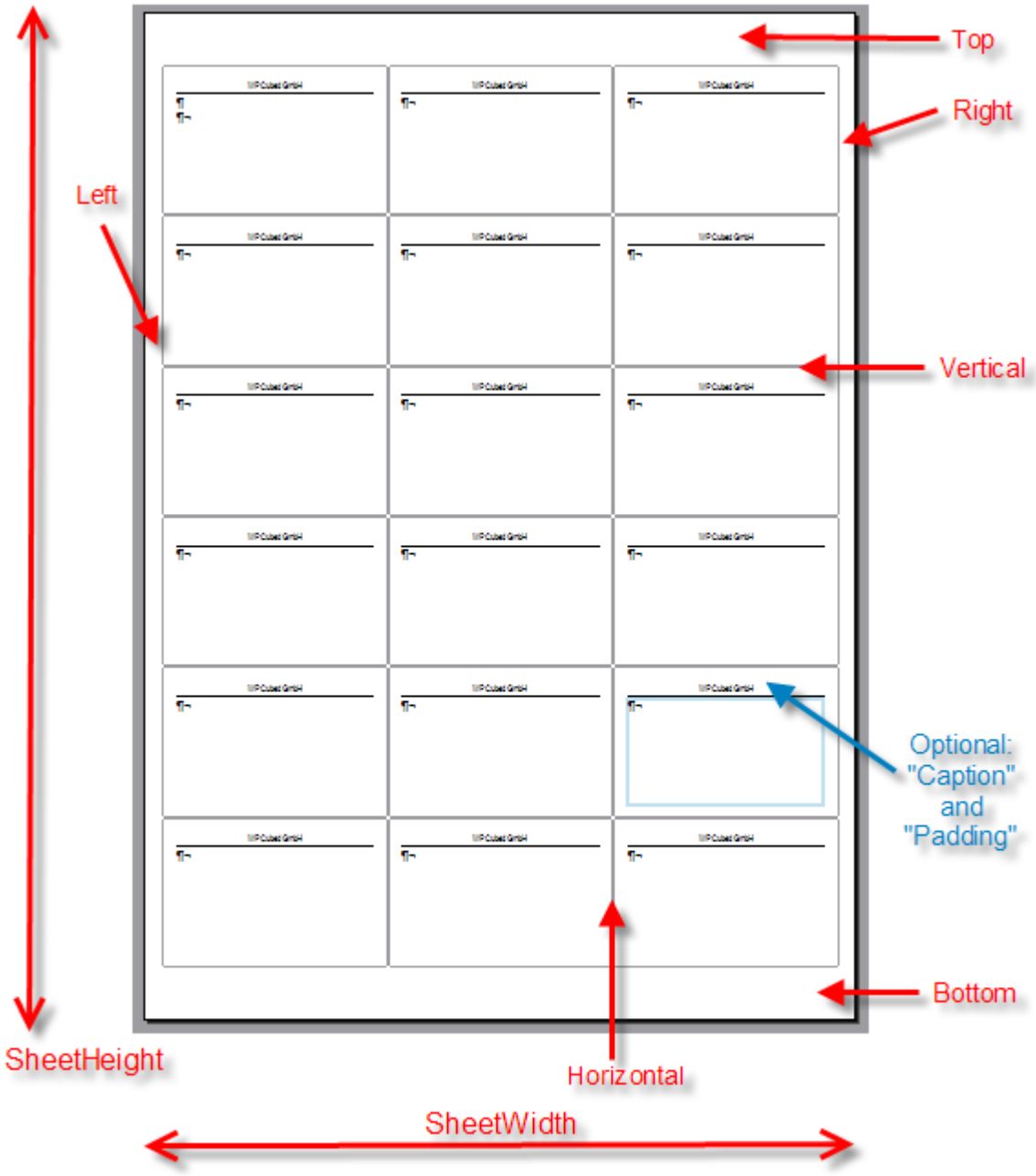
To create the list execute Memo.MergeText for each record and append the mrged text to the editor #2 using Memo2.**AppendOtherText**(0);

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 76.

7.3.5 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 314 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 319 = 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** 316 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 46) 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.

7.3.5.1 C# Code

This C# code uses a loop instead of a database and the event OnFieldGetText 73 is not defined.

```
// Initialize
   IWPMemo memo;
   IWPTextCursor cursor;
   memo = wpdllInt1.Memo;
   cursor = memo.TextCursor;
   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

   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;
```

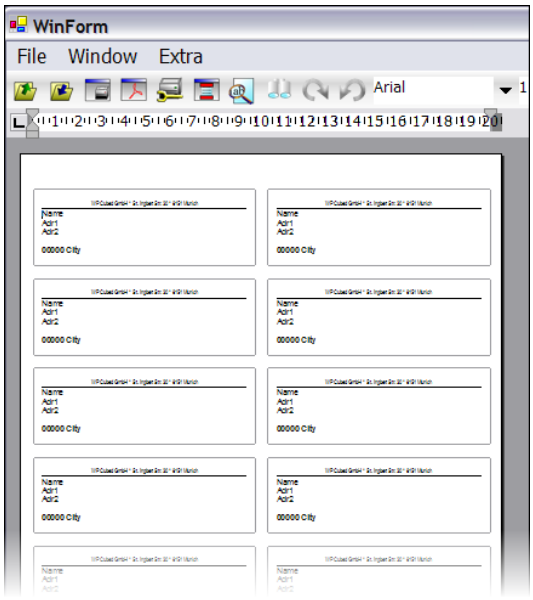
7.3.5.2 VB Code

```
' The interfaces we need
Dim memo As WPDynamic.IWPMemo = Me.WpdllInt1.Memo
Dim cursor As WPDynamic.IWPTextCursor = memo.TextCursor
```

```
' Create the mailing template
memo.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
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
```

7.3.5.3 Resulting Label Sheet

The resulting label sheet is displayed in the editor and can be printed and edited (!)



7.4 HTML/E-MAIL loading and saving

7.4.1 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.

If you want to display a **split editor** with the HTML source (HTML code) and formatted text you can use TextCommandStr 209(5, ..) .

With 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 209(6,1, dll_key)

7.4.2 Create HTML

You can create a HTML file by simply use the Memo.SaveToFile 194 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 150
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 195. 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!

7.4.3 Format Options

Using the format strings 150 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 194 and SaveToString 195)

"\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

7.4.4 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** 319 makes it easy to prepare the e-mail data. You are then free to either use the MapiSendMail function to send it (call Send 325) 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. 321 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 324 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 322, FromAddress 322, Recipients 323, CCList 322, BCCList 322, AttachmentList 321, Body 322.

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();
}
```

7.4.5 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 195 with the format name 150 "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/2wBDAAMCAgMCAgMDAwMEAwMEBQgFBQQEBQoHBwYIDAoM
DAsKCwsNDhIQDQ4RDgsLEBYQERMUFRUVDA8XGBYUGBIUFRT/2wBDAQMEBAUEBQkFBQkUDQsN
FBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBT/wAAR
CAAUABsDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAA
AgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkK
FhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWG
h4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl
5ufo6erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREA
AgECBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYk
NOEl8RcYGRomJygpKjU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6goOE
hYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk
5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwDA+PfxTn8MeHxZ6PNfKz+TbhWj8uHczSZe
NwrEtjgjHQD1rzdfFHxJWGaC71S5uLeaTDrLPKmCCd2wlSF6dOM+lXviV8QYdcu9E0+JZolt
riaYSMzFVwAoO0eucZ9q7y6/an1K00jzZ/ConuHuFURxXW0AfNycg8/LnHvXk11ONnBHs4WW
Gkpe1lZnyvqukaff65c6f5U1ssRM6b0EbyyOcks2B+HHPNLa6DJYwJBJJtdOCJ13OPqe9dnq
et6Rr2sFJZWRkId2hOWjGTnkAE88dO1ePa4b2DWLtDFc4WQhcmfle3f0xXXGTa97c8SpyKXu
6nvvijT4rnxddyuPmYlMKAoABJA4HufzrC1DU2fQ7i+MMXnWsqwouCVIc7SWBPJAHHpRRWb2
FLdGPrF0ugW1q9jbwwzXhaFgDujAzyOAfcg16NpXXhOK7sUml1DUGkZnJPn4z8x9qKKjodF
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/

7.5 Create Text using code

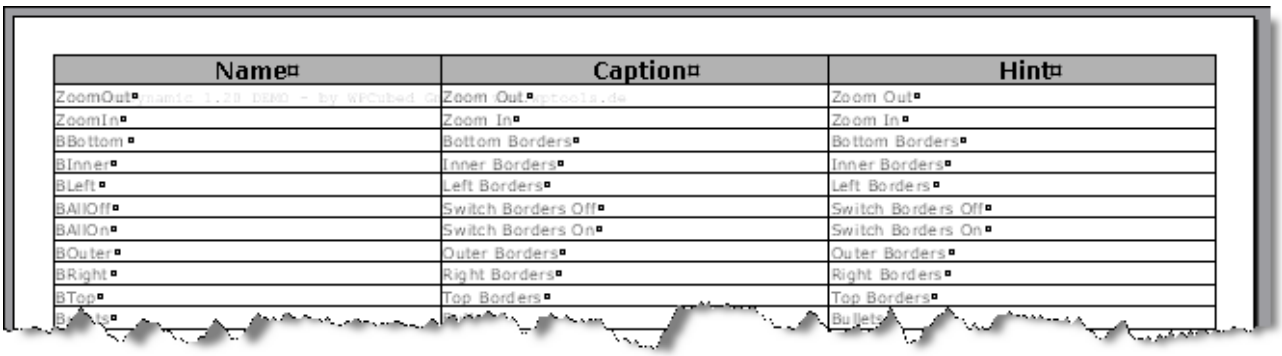
We want to show all the wpa actions nicely formatted in the editor.

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 160 Memo = WPDLLInt1.Memo;
   IWPTextCursor 219 TextCursor = Memo.TextCursor;
   IWPAttrInterface 276 AttrHelper = WPDLLInt1.AttrHelper;
   IWPParInterface 339 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:



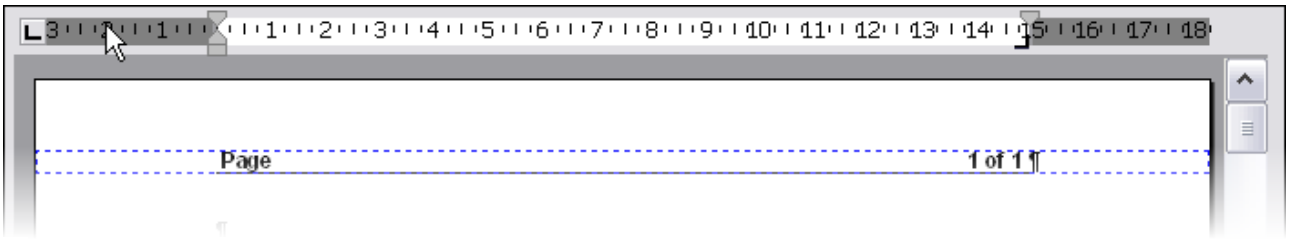
Name	Caption	Hint
ZoomOut	Zoom Out	Zoom Out
ZoomIn	Zoom In	Zoom In
BBottom	Bottom Borders	Bottom Borders
BInner	Inner Borders	Inner Borders
BLeft	Left Borders	Left Borders
BAllOff	Switch Borders Off	Switch Borders Off
BAllOn	Switch Borders On	Switch Borders On
BOuter	Outer Borders	Outer Borders
BRight	Right Borders	Right Borders
BTop	Top Borders	Top Borders
Bullets		Bullets

7.6 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:



7.7 Format C# Code

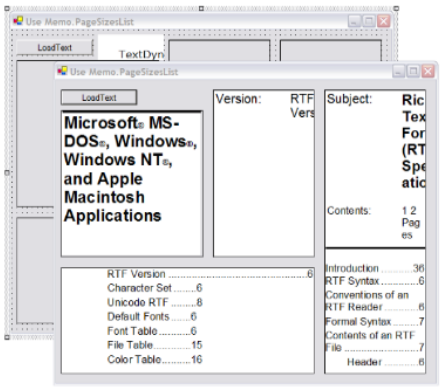
This C# example formats C-Sharp code in a very simple way. It normalizes all paragraphs in the text. It also removes the character attributes from all characters and make the complete text use Courier New, 9pt. The code remove all spaces from the beginning of each paragraph and detect comments and nesting 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 comment
	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);
```

7.8 Extract Metafiles/Print into Rectangles

TextDynamic offers a method to extract each page as a metafile (IWPMemo.GetPageAsMetafile 184).

In combination with the "PageSizesList 167" feature it is possible to let the text "flow" into rectangles:



This C# demo uses an invisible TextDynamic editor and 4 picture boxes. It initializes the editor to use the size of each rectangle as page size and then extract the respective page as metafile.

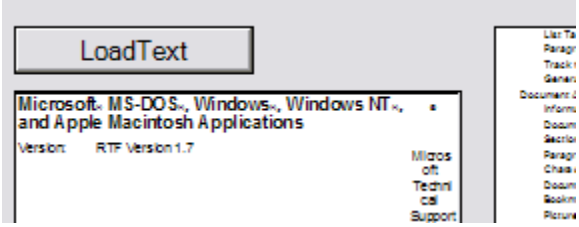
This code initializes the PageSizeList and extracts the metafiles:

```
private void LoadInImage(System.Windows.Forms.PictureBox PictureBox, IntPtr MetaHandle)
{
   if (MetaHandle.ToInt32() !=0)
   {
      Metafile aMetafile=new Metafile(MetaHandle, true);
      PictureBox.Image = aMetafile;
      PictureBox.SizeMode =  PictureBoxSizeMode.StretchImage;
   }
   else PictureBox.Image = null;
}

private void Update()
{
   WPDynamic.IWPMemo memo;
   memo = wpdllInt1.Memo;
   memo.PageSizeList.Clear();
   memo.PageSizeList.Active = false;
   memo.PageSizeList.Resolution = 120;
   memo.PageSizeList.Add(pictureBox1.Width,pictureBox1.Height);
   memo.PageSizeList.Add(pictureBox2.Width,pictureBox2.Height);
   memo.PageSizeList.Add(pictureBox3.Width,pictureBox3.Height);
   memo.PageSizeList.Add(pictureBox4.Width,pictureBox4.Height);
   memo.PageSizeList.Active = true;

   // Load the images
   LoadInImage(pictureBox1, memo.GetPageAsMetafile(0,4));
   LoadInImage(pictureBox2, memo.GetPageAsMetafile(1,4));
   LoadInImage(pictureBox3, memo.GetPageAsMetafile(2,4));
   LoadInImage(pictureBox4, memo.GetPageAsMetafile(3,4));
}
```

The "load text" button is connected to this simple method:

```
private void LoadText_Click(object sender, System.EventArgs e)
{
   WPDynamic.IWPMemo memo;
   memo = wpdllInt1.Memo;
   memo.LayoutMode = WPDynamic.LayoutMode.wplayFullLayout;
   memo.AutoZoom = WPDynamic.AutoZoom.wpAutoZoomOff;
   if(memo.Load("","")) Update();
}
```

Of course it is possible to display scaled text:



simply change the reference resolution value:

```
memo.PageSizeList.Resolution = 40;
```

Note: The TextDynamic demo version will print a red cross.

7.9 Reporting

7.9.1 Introduction

TextDynamic contains an optional reporting feature which is controlled by the interface IWPReport 372.

There are plenty reporting applications and tools available so you may ask why a reporting engine has been implemented into a word processing component?

1) 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.

2) 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.

3) 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) Many reporting tools are very complicated to use or do not offer end user modifications at all.

With TextDynamic we tried to avoid problems at the end user side by splitting up the report logic into two parts.

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.
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.

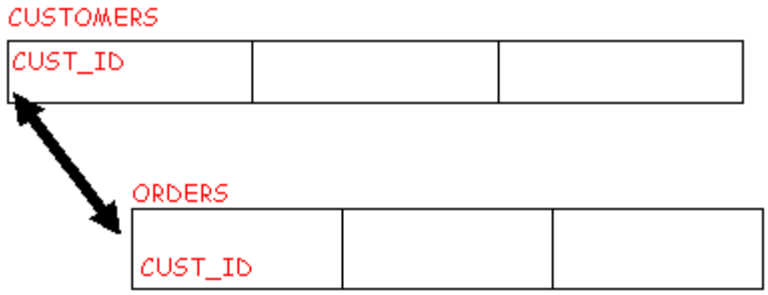
7.9.2 Reporting Basics

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 72. 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. 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 seperating the necessary tasks into several steps.

(a) 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.

From the structure created in (a) TextDynamic can create a raw reporting template. You 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.

(b) 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)*

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.

7.9.2.1 Collect groups and fields

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 57 ".\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 53 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 372
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 goup 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.

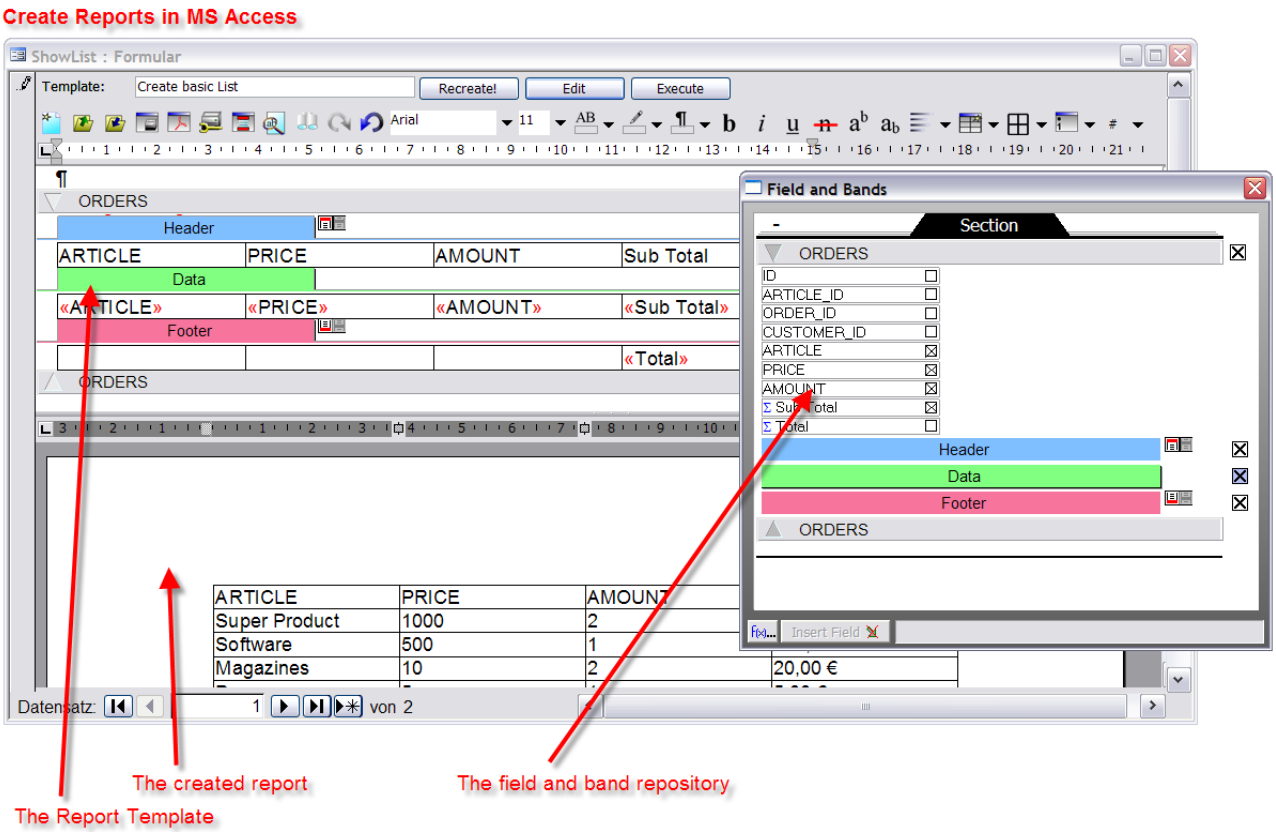
To show the report editor without recreation of the template use this code:

```
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
```



7.9.2.2 Open Query and Control Reporting

To completely control the report creation use the events of TextDynamic. There are 3 events, one controls the grop processing, one reads the values of fields and one reads out variables.

7.9.2.2 A) Event driven reporting

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.

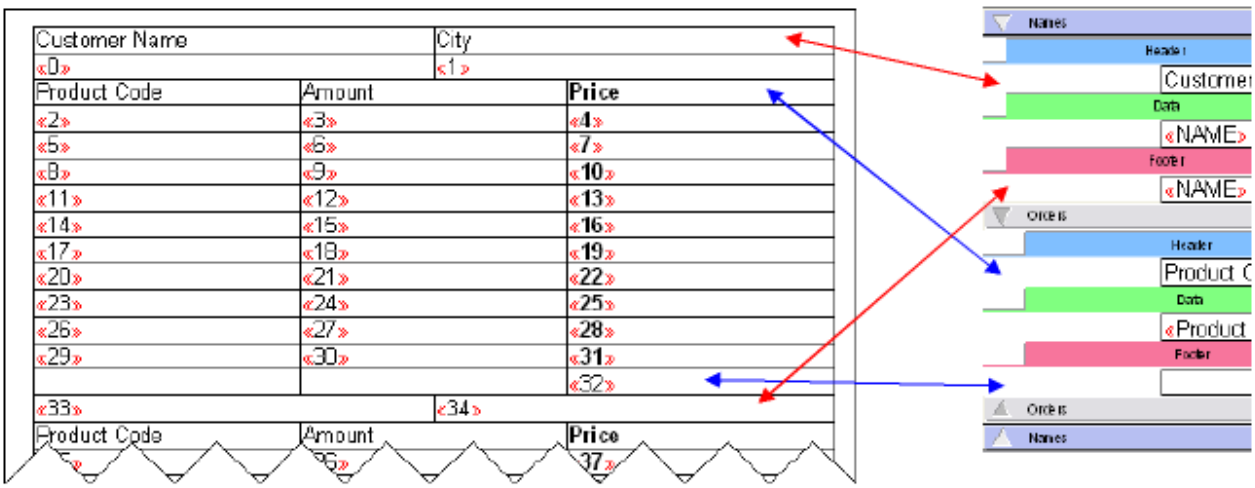
```
private void wpdllInt1_OnReportState(
        object Sender, string Name,
        int State, WPDynamic.IWPReportBand Band,
        ref bool Abort)
{
        if (State==WPDynamic.commands.REP_BeforeProcessGroup)
                                Abort = Band.Count>10;
        else Abort = false;
}
```

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



In the following chapter we show some real word MS Access code.

7.9.2.2 B) MS Access code

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
```

7.9.3 API

Interfaces

IWPReport 372
IWPReportBand 376
IWPReportVar 384

Events

OnFieldGetText 129
OnReadFormulaVar 132
OnReportState 133

Methods

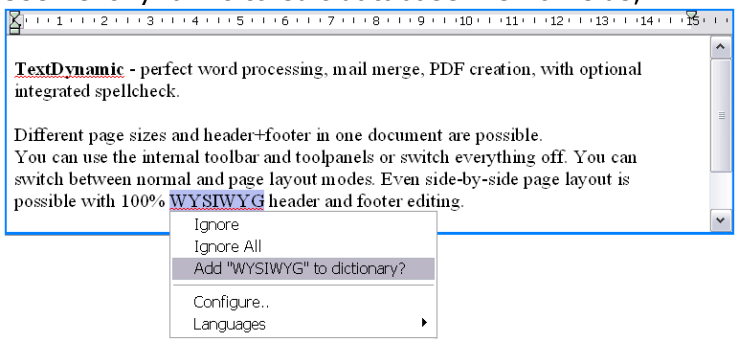
WPDLLInt.CreateReport 110
WPDLLInt.Report 113
IWPTextCursor.ReportConvertTable 257
IWPTextCursor.ReportConvertText 258
IWPTextCursor.ReportInputBand 258
IWPTextCursor.ReportInputGroup 258

8 API Reference

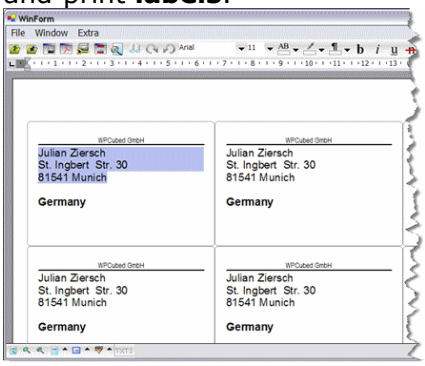
8.1 WPDLLInt

Description
The WPDLLInt object is the central part of the TextDynamic word processing component.

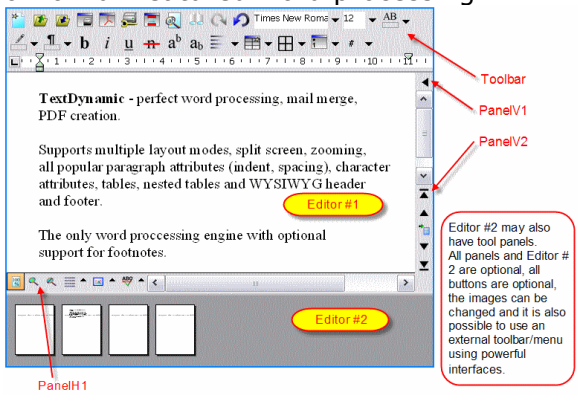
Use TextDynamic to edit database memo fields,



you can use Memo.LabelDef 314 to quickly create, edit and print **labels**.



or for full featured word processing:



Work with the powerful mail merge 189, create documents under program control using the TextCursor API 219 or the optional reporting 372.

Please also see the topic "Configure the Editor 51" under "Getting Started 18"!

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 271
To prepare and send e-mails use the property Mapi 319.
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.

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 100.

When using .NET you can please use the static method SetDLLName(string) 100 as early as possible in the program:

```
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 160, IWPTextCursor 219, ...) 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 112;
IWPTextCursor TextCursor = Memo.TextCursor;

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

wpdllint1.ReleaseInt 123(TextCursor); // Please use ReleaseInt to avoid problems caused by the garbage collector
wpdllint1.ReleaseInt 123(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 116 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 114 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 110.

The following interfaces are supported by TextDynamic:

Name	Published by	Notes
IWPMemo 160	Memo Memo2 CurrMemo	Main editing interface

IWPTextCursor 219	TextCursor IWPMemo 160.TextCursor	Create text by code, move cursor
IWPPdfCreator 271	PdfCreator	create PDF files
IWPSpell 388	IWPMemo 160.SpellCtrl	Modify the setup of the optional spellcheck engine.
IWPMAPI 319	IWPMemo 160.MAPI	reserved to create and and send e-mails.
IWPPageSize 333	PageSize IWPMemo 160.PageSize	Change document page size
IWPTextObj 396	CurrObj CurrSelObj IWPMemo 160.CurrObj IWPMemo 160.CurrSelObj Contents.EmbeddedObject 312 Contents.FieldObject 313 CurrPar.CharObj 353 Events: OnFieldEnter OnFieldLeave OnHyperlink OnTextObjectMouse OnLoadExtImage OnBeforeSaveImage OnAfterSaveImage OnTextObjectGetText OnEnumTextObj	Modify images and other objects, such as hyperlink and mail-merge tags.
IWPParInterface 339	CurrPar IWPMemo 160.CurrPar	Modify current paragraph. Change text and set paragraph properties.
–	CurrStyle IWPMemo 160.CurrStyle	Modify current style to change paragraph properties.
IWPAttrInterface 276	IWPMemo 160.CurrAttr	Modify the current writing mode or change the selected text.
–	CurrParAttr IWPMemo 160.CurrParAttr Event OnEnumParOrStyle Event OnCreateNewCell	Modify font attributes of all characters in current paragraph.
–	CurrStyleAttr IWPMemo 160.CurrStyleAttr	Modify font attributes defined by current style.
–	IWPFieldContents 307. FieldAttr	Modify font attributes of mail merge field.
–	IWPParInterface 339.CharAttr ()	Modify font attributes of certain character in paragraph.
–	IWPMemo 160.CurrAttr	Modify the current writing mode
–	CurrStyleAttr IWPMemo 160.CurrStyleAttr	Modify font attributes defined by current style.
IWPPrintParameter 370	IWPMemo 160. PrintParameter	Modify print options.
IWPDataBlock 296	Memo.BlockFind 178 Memo.BlockAdd 176 Memo.ActiveText 162 Event OnEnumDataBlocks	Manage header and footer texts.
IWPFieldContents 307	Event OnFieldGetText	Mailmerge - replace fields with text or images.
IWPDllButton 303	Event OnButtonClick	Provide code for custom actions added to internal toolbars.
IWPCharacterAttr 292	Memo.SpecialTextAttr() 169	Modify appearance of hyperlinks and other "special" text.
IWPReport 372	Property Report	Create and manage a report pre-template. Show report template editor.
IWPLabelDef 314	Property Memo.LabelDef	Edit, preview and print mailing labels.

8.1.1 Properties

8.1.1.1 AttrHelper

Applies to
WPDLLInt 94

Declaration
```
property AttrHelper: IWPAttrInterface 276 read Get_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 = wpdllInt1.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 146

8.1.1.2 BorderWidth

Applies to
WPDLLInt 94

Declaration
```
property BorderWidth: Integer read Get_BorderWidth write Set_BorderWidth;
```

Description
This is the width of the margin between the outer border of the control window and the embedded editor and GUI elements.

8.1.1.3 CurrAttr

Declaration
```
IWPAttrInterface 276 CurrAttr;
```

Description
This is the interface to the current writing attribute of the active editor. If you need to change the selected text or, if no text is selected, the current writing mode use property TextAttr 104.

Please note that a click in the editor will reset the writing mode to the mode used by the text at the cursor position. If you change CurrAttr in code you need to set the focus into the editor after the update.

8.1.1.4 CurrObj

Applies to
WPDLLInt 94

Declaration
```
property CurrObj: IWPTextObj 396 read Get_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.
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);
```

8.1.1.5 CurrPar

Declaration
```
property CurrPar: IWPParInterface 339 read Get_CurrPar;
```

Description
This is the interface to manipulate the current paragraph. This is the paragraph the cursor (insertion marker) is located within.
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 98. If you only need to change the attribute of certain characters use either CharAttr 352 or SetCharAttr 365.

This interfaces is also provided as IWPMemo.CurrPar 164.

Important: The interfaces CurrPar and CurrParAttr 98 always access the current cell/paragraph - they cannot be used to edit a cell/paragraph which lost focus.

Example:

```
IWPParInterface par = wpdllInt1.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);
```

Tip: You can use paragraph id values in SetPtr 366 to set the current paragraph reference to a different pa

8.1.1.6 CurrParAttr

Applies to
WPDLLInt 94

Declaration
```
property CurrParAttr: IWPAttrInterface 276 read Get_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 use either CharAttr 352 or SetCharAttr 365.
This interfaces is also provided as IWPMemo.CurrParAttr.

Example: The current paragraph should be bold and use the font "Times New Roman".

```
IWPAttrInterface parattr = wpdllInt1.CurrParAttr;
parattr.IncludeStyles(1);
parattr.SetFontface("Times New Roman");
```

8.1.1.7 CurrSelAttr

Applies to
WPDLLInt 94

Declaration
IWPAttrInterface 276 CurrSelAttr;

Description
This interface can be used to change the attributes of the text which is currently selected.

8.1.1.8 CurrSelObj

Applies to
WPDLLInt 94

Declaration
IWPTextObj 396 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");
}
```

8.1.1.9 CurrStyle

Applies to
WPDLLInt 94

Declaration
property CurrStyle: IWPParInterface 339 **read** Get_CurrStyle;

Description
This property can be used to manipulate the current style. This is the style which has been added last or which was located by function SelectStyle 196.

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 propereties of a paragraph and assign to a style and vice versa.

Category
Paragraphstyle Support 157

8.1.1.10 CurrStyleAttr

Applies to
WPDLLInt 94

Declaration
```
property CurrStyleAttr: IWPAttrInterface 276 read Get_CurrStyleAttr;
```

Description
This interface is used to change the character attributes defined by the current style. Also see CurrStyle 99.

Category
Paragraphstyle Support 157

8.1.1.11 DLLName

Applies to
WPDLLInt 94

Declaration
```
property DLLName: WideString read Get_DLLName write Set_DLLName;
```

Description
When using the OCX please specify the path to the DLL in property DLLName. When using .NET please use the static method SetDLLName, i.e.

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

Once the name was applied it should not be changed anymore. It is possible to read the name from the registry: Specify the path to the registry key (type string) preceeded by {hkcu} to use HKEY_CURRENT_USER, or {hklm} to use HKEY_LOCAL_MACHINE.

Example: "{hkcu}Software\MyCompany\TextDynamic\path" will load the name from the string property path under HKEY_CURRENT_USER\Software\MyCompany\TextDynamic.

Note: The key Software\WPCubed\TextDynamic\path is created by the TextDynamic setup script and **must not** be used in your application. We use it our example projects only when the property DLLName has its default value "{please_change_this}";

8.1.1.12 Enabled

Applies to
WPDLLInt 94

Declaration
```
property Enabled: WordBool read Get_Enabled write Set_Enabled;
```

Description
If this property is false the control will not accept any input (keyboard or mouse)

8.1.1.13 EventBand

Applies to

WPDLLInt 94

Declaration
```
property EventBand: IWPReportBand 376 read Get_EventBand;
```

Description
This property only exists in the **ActiveX**. It had been added to provide access to the current band inside the event OnReportState (used for reporting) only in case the development system does not pass reference parameters as expected.

8.1.1.14 EventButton

Applies to
WPDLLInt 94

Declaration
```
property EventButton: IWPDllButton 303 read Get_EventButton;
```

Description
This property only exists in the **ActiveX**. It had been added to provide access to the current button inside the event OnButtonClick (used for custom buttons on toolbar) only in case the development system does not pass reference parameters as expected.

8.1.1.15 EventField

Applies to
WPDLLInt 94

Declaration
```
property EventField: IWPFieldContents 307 read Get_EventField;
```

Description
This property only exists in the **ActiveX**. It had been added to provide access to the current field and field contents inside the event OnFieldGetText (used by mail merge) only in case the development system does not pass reference parameters as expected.

8.1.1.16 InitScriptXML

Applies to
WPDLLInt 94

Declaration
```
property InitScriptXML: WideString read Get_InitScriptXML write
Set_InitScriptXML;
```

Description
This is used to assign a set of properties using XML script. It is redundant because of the methods SetEditorMode and SetLayout.

8.1.1.17 MAPI

Applies to
WPDLLInt 94

Declaration

```
property MAPI: IWPMapi 319 read Get_MAPI;
```

Description
This property provides access to the IWPMapi interface. This interface will, be used to create and send e-mails. Currently it is not used.

8.1.1.18 PageSize

Applies to
WPDLLInt 94

Declaration
```
property PageSize: IWPPageSize 333 read Get_PageSize;
```

Description
You can use this interface to change the current paper size. Please note that all values are twip values. 1 inch = 1400 twip.

8.1.1.19 PDFCreator

Applies to
WPDLLInt 94

Declaration
```
property PDFCreator: IWPPdfCreator 271 read Get_PDFCreator;
```

Description
This interface makes it possible to access the internal PDF creation engine - if you have the license to use it. The methods Print and PrintSecond are used to export directly to PDF. Please note that unless you have the "server" license a file open dialog will always be displayed, even if thge property PDFFilename was set. This file open dialog will be displaied agian when the file was locked to give the user a chance to select a different file or to close the application which is using this file.

8.1.1.20 Readonly

Applies to
WPDLLInt 94

Declaration
```
property Readonly: WordBool read Get_Readonly write Set_Readonly;
```

Description
If this property is tre the editor #1 does not allow modifications by the user.

8.1.1.21 Readonly2

Applies to
WPDLLInt 94

Declaration
```
property Readonly2: WordBool read Get_Readonly2 write Set_Readonly2;
```

Description

If this property is tre the editor #2 does not allow modifications by the user.

8.1.1.22 ShowHints

Applies to
WPDLLInt 94

Declaration
```
property ShowHints: WordBool read Get_ShowHints write Set_ShowHints;
```

Description
If this property is true, the editor will disply fly over hints for the toolbar icons.

8.1.1.23 SpellCtrl

Applies to
WPDLLInt 94

Declaration
```
property SpellCtrl: IWPSpell 388 read Get_SpellCtrl;
```

Description
This property provides access to the interface IWPSpell 388 which can be used to change the setup of the spellcheck controller. Please modify the setup parameter before you call SetEditorMode to avoid that the default setup is used.

Note

You can also use the method CommandEx 107(9500, n, strparam) to execute some of the methods. This may be useful if you cannot access the interface in your developing language.

8.1.1.24 Text

Applies to
WPDLLInt 94

Declaration
```
String Text;
```

Description
This property is bindable. It can be used to load and save the text in the upper editor. The format which is used for loading and saving is selected by property TextFormat 104.

When using the **ActiveX** the datatype of property Text and Text2 is VARIANT - you can assign a usual unicode string and a variant byte array. When reading this propetry always(*) a variant byte array will be created. To get a standard string from the editor use the method SaveToString 195. To get a variant array of bytes you can use SaveToVar 196. To load/insert from a string or a variant array use LoadFromVar 188. (The method LoadFromString 188 is similar but will only accept strings)

When binding the editor to a database use a binary field to hold the data. The field should be an image field in SQL server, or an oleobject field in MS Access.

*) If you append "-useolestring," to the property TextFormat, a standard double byte OLE string instead of the variant array of byte will be created.

Category
Load and Save 150

8.1.1.25 Text2

Applies to
WPDLLInt 94

Declaration
```
String Text2;
```

Description
This property is bindable. It can be used to load and save the text in the lower editor. The format which is used for loading and saving is selected by property TextFormat 104.

Category
Load and Save 150

8.1.1.26 TextAttr

Applies to
WPDLLInt 94

Declaration
IWPAttrInterface 276 TextAttr;

Description
If you need to change the selected text or, if no text is selected, the current writing mode use property TextAttr. You can use it to create your own toolbar, 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.

Please note that a click in the editor will reset the writing mode to the mode used by the text at the cursor position. If you change TextAttr in code you need to set the focus into the editor after the update of the current writing mode.

8.1.1.27 TextCursor

Applies to
WPDLLInt 94

Declaration
IWPTextCursor 219 TextCursor;

Description
This is the interface to the current cursor object. The cursor object contains most methods to create text and the properties which read and modify the current position in the text.

8.1.1.28 TextFormat

Applies to
WPDLLInt 94

Declaration
```
String TextFormat;
```

Description
This property select the format used by the properties Text and Text2. You can use the strings "ANSI", "RTF", "HTML" to select the major file formats but also "UNICODE" and "WPT". The WPT format is our proprietary fileformat which is also used by the product WPTools.

If not format or "AUTO" was specified the reading process will check the format automatically, this works for RTF, HTML and WPT format. All other text will be loaded as ANSI.
After the name of the format options are possible, for example "RTF-ignorepagesize". See http://www.wpcubed.com/manuals/formatstr.htm.

Category
Load and Save 150

8.1.1.29 Version

Applies to
WPDLLInt 94

Declaration
```
int Version;
```

Description
This is the version of the OCX.

8.1.1.30 VersionDLL

Applies to
WPDLLInt 94

Declaration
```
int VersionDLL;
```

Description
This is the version of the loaded editor kernel.

8.1.1.31 AllowDrop

This property is only used by the .NET component to switch drag&drop support on.

For .NET drag&drop example code please see Memo.GetPosAtXY 185.

If you have several editors on your form drag&drop between this editors is automatically handled! You can disable it using Memo.SetBProp 197 - ClipBoardOptions.

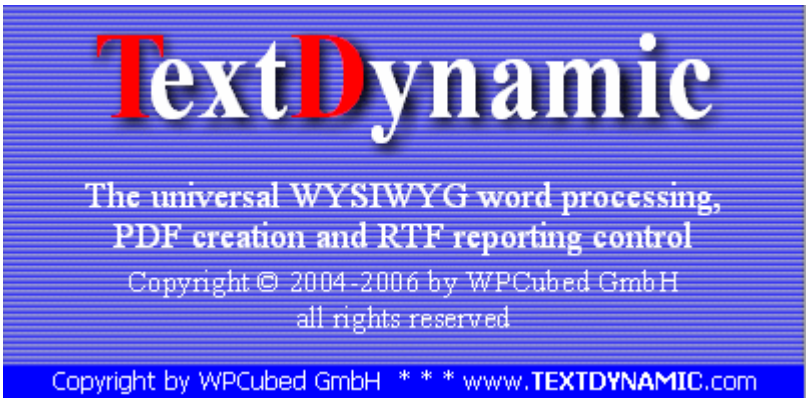
8.1.2 Methods

8.1.2.1 AboutBox

Applies to
WPDLLInt 94

Declaration
```
procedure AboutBox;
```

Description
This methods shows an information about the DLL.



8.1.2.2 BeginPrint

Applies to
WPDLLInt 94

Declaration
```
procedure BeginPrint(const Title: WideString);
```

Description
This method can be used to open a printing cue. Now you can execute Memo.Print and all printing output will go into the same cue.
Please call EndPrint 106 when done.

Note:
You can alternatively use Memo.TextCommandStr 209 ID 13 and 14 (Command ID 13 - BeginPrint 212, Command ID 14 - EndPrint 212)

Category
Printing 156

8.1.2.3 EndPrint

Applies to
WPDLLInt 94

Declaration
```
procedure EndPrint;
```

Description
This procedure closes a printing cue started with BeginPrint.

Category
Printing 156

8.1.2.4 Clear

Applies to
WPDLLInt 94

Declaration
```
procedure Clear;
```

Description
This method clears both editors completely. It will also clear the cached attribute records and

the styles.

8.1.2.5 CommandEx

Applies to
WPDLLInt 94

Declaration
```
function CommandEx(ComID: Integer; param: Integer; const StrParam: WideString): Integer;
```

Description
This method can be used to execute a custom command which requires a string parameter.

Also see Command 107.

8.1.2.6 Command

Applies to
WPDLLInt 94

Declaration
```
int Command(int com, int param);
```

Description
This method can be used to execute a custom command. It is reserved for future and custom extensions.
Please also see CommandEx 107 - since both methods use the same IDs. In the .NET we have created several overloaded methods but in the OCX you have to use Command or CommandEx depending on the parameter types you need to specify.

There are several groups of commands: Security 107, change toolbar design 107, custom spellcheck 108.

8.1.2.6 A) Security

WPDLL_COM_PROTECTEDTEXT = 9119:
Set global security options - works like Memo.SetBProp(14, param, 1) 206.

8.1.2.6 B) Change Toolbar Design

WPDLL_COM_DialogBackground = 9501
parameter=1 activates a brushed metal background for the dialogs.

You can also load a background image for the tiling

```
wpdllInt1.Command(9501,1,"c:\\someimage.bmp");
```

WPDLL_COM_ToolbarDesign = 9502
Select the design style for all toolbars which do not use the <design> tag in the XML. If the command does not show any effect please check the layout description with the WPImagePack application and disable the tags <design.../>.

This modes are supported:

WPDLL_COM_ToolBarColorFrom = 9503
Change the start color for the gradient fill used by mode 3. Default is $FFFFFF

WPDLL_COM_ToolBarColorTo = 9504
Change the end color for the gradient fill used by mode 3. Default is $FE9696

```
C# Example:
wpdllInt1.Command(9502,3,0);
wpdllInt1.Command(9503,wpdllInt1.ToRGB(Color.LightBlue),0);
wpdllInt1.Command(9504,
wpdllInt1.ToRGB(Color.Blue),0);
```

WPDLL_COM_SetHorzBack = 9505
Load a bitmap which is stretched on the background of horizontal toolbars. You can realize individual shading using such a bitmap.

WPDLL_COM_SetVertBack = 9506
Load a bitmap which is stretched on the background of vertical toolbars.

8.1.2.6 C) Custom Spellcheck

WPDLL_SetCUSTOMSPELL_EVENT = 907

"Custom spell check" allows it to use an external data base with words to verify the spelling. We recommend to use the integrated spell check 160 methods instead, because they guarantee high performance and compact dictionaries. (Requires optional "spell" license)

But if you need to use an existing spell check solution, TextDynamic allows this, too:

Use a parameter=0 to disable **custom** spellcheck, use a parameter = A (any number) to enable it.

You should call Command(907) before SetEditorMode.

Please note, if you enable the "custom" spell check, the internal spell check feature is disabled!

The event OnEnumParOrStyle 127 will be triggered to check words with EventParam=A. You have to change the variable Abort=true in case the word is wrong. You can use GetSubText 356 to extract this word.

```
// Initialization
...
WPDLLInt1.Command(907,1);
// Select Editor Mode !
```

```
WPDLLInt1.SetEditorMode( ...

// The event handler
private void WPDLLInt1_OnEnumParOrStyle(object Sender, bool IsControlPar, int StartPos,
   int Count, WPDynamic.IWPParInterface ParText,
   WPDynamic.IWPAttrInterface ParAttr, int EventParam,
   ref bool Abort)
{
   if (EventParam==1)
   {
      if(ParText.GetSubText(StartPos,Count) == "right")
           Abort = false;
      else Abort = true;
   }
}
```

Please see the example at OnMouseDownWord 139. This event can be used to create a popup menu. If custom spell checking was activated bit 10 will be set in parameter Shift if the word was marked to be wrong.

You can use the Memo.TextCommand 208(7,ParamA,0) to start and stop spell check.

The following values are allowed for ParamA:

0= Start SpellCheck (using the optional internal spellchecker),
1= Start Thesuarus (reserved),
2= Start SpellAsYouGo (internal or external),
3= Stop SpellAsYouGo, (internal or external),
4= Show SpellCheckSetup (internal only).

```
// Start SpellAsYouGo
WPDLLInt1.Memo.TextCommand(7,2,0);

// Stop SpellAsYouGo
WPDLLInt1.Memo.TextCommand(7,3,0);
```

With the method SetIProp 206 it is possible to change the spell check strategy.

Memo.SetIProp(4,0) : Words are checked when the paragraph is initialized. The complete text is checked after loading
Memo.SetIProp(4,1) : (default) Words are checked before a paragraph is painted.
Memo.SetIProp(4,2) : - check whole text first

Only in case the IWPSpell interface SpellCtrl 388 cannot be used, use the following command 9500 to control in integrated spell checker:

WPDLL_COM_SPELLCTRL = 9500

Param=1: ClearAll - Remove all references to dictionaries
Param=2: Load setup from INI file specified by strparam
Param=3: Save setup to INI file specified by strparam
Param=4: Load setup from REGISTRY specified by strparam
Param=5: Save setup to REGISTRY specified by strparam
Param=6: AddFromPath - Load all DCT files from path specified by strparam
Param=7: AddFromFile - Load the DCT file specified by strparam
Param=8: UserDictAdd - set user dictionary to file
Param=9: UserDictRemove - set user dictionary to file
Param=10: SetLanguage - activate the language specified by strparam
Param=11: Execute command - editor 1, mode = strtoint(strparam)
Param=12: Execute command - editor 2, mode = strtoint(strparam)

8.1.2.6 D) Background Image

The editor has the possibility to display a background images on each page. There can be a left and a right image.

The left image is tiled in vertical direction, the right image is tiled in both directions. Both images can be used to mimic a paper notebook.

This command can be used to load the respective Bitmap (BMP) files:

CommandEx 107 141, 0, image_path loads the left image
CommandEx 142, 0, image_path loads the right image

Also available is

Command 143, 0, image_path

This loads the watermark image which will be displayed on the page without tiling.

All this images can be activated and deactivated using the WPA actions "ShowWatermark" and "ShowBackgroundImage".

8.1.2.7 CreateReport

Applies to
WPDLLInt 94

Declaration
```
procedure CreateReport;
```

Description
Using CreateReport the reporting is started. The template must be loaded into Editor #1. Editor #2 will be used for the resulting text.

Category
Reporting 94

8.1.2.8 CurrMemo

Applies to
WPDLLInt 94

Declaration
```
function CurrMemo: IWPMemo 160;
```

Description
This is the IWPMemo interface of the active editor. "Active" is the editor which had the focus last. Using SelectEditor you can activate editor 1 or 2.

8.1.2.9 EditorStart

Applies to
WPDLLInt 94

Declaration
```
procedure EditorStart(const licensename: WideString; const licensekey:
WideString; licensecode: Integer);
```

Description

This method is used to iniztialize the editing. You need to pass the licensing information.

8.1.2.10 GetInterface

Applies to
WPDLLInt 94

Declaration
```
function GetInterface(Editor: Integer; Select: Integer): IUnknown;
```

Description
This method can be used to retrieve a custom interface. (Reserved for future extensions)

8.1.2.11 GetItemList

Applies to
WPDLLInt 94

Declaration
```
function GetItemList(Editor: Integer; ListID: TxItemListIDS; Quoted: WordBool;
const Delimiter: WideString): WideString;
```

8.1.2.12 GetPrinterList

Applies to
WPDLLInt 94

Declaration
```
string GetPrinterList;
```

This method returns a string with a list of printer names. The name can be used with SelectPrinter 114 or Memo.TextCommandStr(10, name). 212

8.1.2.13 GetText

Applies to
WPDLLInt 94

Declaration
```
function GetText(Editor: Integer; const Format: WideString; OnlySelection:
WordBool): WideString;
```

Description
This method converts the loaded text into a string - using a special format. You can select the type ("RTF") and also append certain options, such as "RTF-onlybody" if you want to save the RTF text but no header or footer. (see: http://www.wpcubed.com/manuals/formatstrings.htm)

Category
Load and Save 150

8.1.2.14 GetTextVar / GetBytes

Applies to
WPDLLInt 94

Declaration
```
object GetTextVar(int Editor, string Format,  bool OnlySelection);
```

The .NET Assembly also defines:
public byte[] GetBytes(int Editor, bool OnlySelection, string Format)

Description
This method can be used (only in the OCX) to save the text to a varaiant. This has the advantage that RTF data (which is plain ASCII code) does not have to be passed as unicode string.

Parameters	
Editor	The editor to update: 1 or 2
Data	The text as OleVariant - an array of bytes will be created.
Format	A string which specifies the saving options.
OnlySelection	true to only save the selected text.

Returns
True if saving went ok.

Also see SetTextVar 119!

8.1.2.15 Memo

Applies to
WPDLLInt 94

Declaration
```
function Memo: IWPMemo 160;
```

Description
This is the IWPMemo interface of editor #1. This interface contains all important methods and properties to access interfaces such as TextCursor 219.

RTF2PDF Note:
Our ASP product RTF2PDF also publishes this property. It gives you access to the IWPEditor interface which includes most of the methods and properties of IWPMemo. We recommend to assign the value of this property to a variable, so, when you decide to create an ASP project from your application, you only need to change one line. All other important interfaces are available inside of the "Memo" object (TextCursor, CurrPar, CurrAttr etc)!

TextDynamic:
```
IWPMemo Memo;
Memo = wpdllint1.Memo;
// used Memo ....
```

RTF2PDF:
```
IWPEditor Memo;
Memo = rtf2pdf1.Memo;
// used Memo ....
```

.NET: We recommend to assign the value of "Memo" to a variable and call wpdllint.ReleaseInt 123(variable) when the reference is not any longer required.

8.1.2.16 Memo2

Applies to
WPDLLInt 94

Declaration
```
function Memo2: IWPMemo 160;
```

Description
This is the IWPMemo interface of editor #2. This interface contains all important methods and properties to access interfaces such as IWTextCursor.

Our ASP product RTF2PDF also publishes this property.

8.1.2.17 ReadRecentExceptions

Applies to
WPDLLInt 94

Declaration
```
function ReadRecentExceptions(ClearList: WordBool): WideString;
```

Description
This function can be useful for debugging purpose.
It can be used to extract a list with all recent internal exception messages. The string will be empty if no errors took place. If the boolean parameter ClearList was passed as "true", the internal list will be cleared. Note that the provided list contains the error messages collected from all active TextDynamic controls.

8.1.2.18 Report

Applies to
WPDLLInt 94

Declaration
```
function Report: IWPReport 372;
```

Description
This method allows the access to the interface IWPReport 372.
Please don't forget to activate the double editor and the reporting support using SetEditorMode(EditorMode.wpmodDoubleEditor, EditorXMode.wpmodexReporting| ...)

Category
Reporting 94

8.1.2.19 SelectEditor

Applies to
WPDLLInt 94

Declaration
```
procedure SelectEditor(EditorNr: Integer; SetFocus: WordBool);
```

Description
Using this function you can select the active editor, 1 or 2.

8.1.2.20 SelectPrinter

Applies to
WPDLLInt 94

Declaration
```
int SelectPrinter(string Name);
```

Description
You can select the current printer by passing the name of the printer. A list of names is provided by GetPrinterList 111.

Another option to select a printer is Memo.TextCommandStr(10, name) 212.

Category
Printing 156

8.1.2.21 SetDisableFlags

Applies to
WPDLLInt 94

Declaration
```
procedure SetDisableFlags(Editor: Integer; Flags: Integer);
```

Description
Using the method SetEnableFlags it is possible to modify the editor #1 or #2:
Certain options which are active by default can be deactivated:

```
  wpoptNoEdit      =   1;  // editor is readonly
  wpoptNoSelect    =   2;  // selection not visible
  wpoptNoFileDrop  =   4;  // Don't accept image files from Explorer
  wpoptNoDragDrop  =   8;  // Disable Drag&Drop
  wpoptNoCopy      =  16;  // Disable Copy
  wpoptNoPaste     =  32;  // Don't allow any paste
  wpoptNoPasteIMG  =  64;  // Don't allow paste of images
  wpoptNoPasteRTF  = 128;  // Don't allow pase of RTF (or HTML)
  wpoptNoEditHeader= 512;  // Don't switch to header/footer to edit
  wpoptNoResizeImages  = 1024; // User cannot resize images
  wpoptNoResizeTables  = 2048; // User cannot move or resize tables
  wpoptNoResizeColumns = 4096; // User cannot resize columns
  wpoptNoResizeRows    = 8192; // User cannot resize Rows
```

Note: The interface IWPMemo also includes the method SetBProp - this method allows it to set docends of other flags. Please see online reference.

8.1.2.22 SetEditorMode

Applies to
WPDLLInt 94

Declaration
```
procedure SetEditorMode(Mode: Integer; XMode: Integer; GUI1: Integer; GUI2:
Integer);
```

Description
TextDynamic can work in several different ways. The mode is selected by using the method WPDLLInt1.SetEditorMode(). One Control can host two editors at once, for example to support split screen: the start and the end of a longer text is displayed at the same time.

```
WPDLLInt1.SetEditorMode(
 EditorMode.wpmodSingleEditor,
 EditorXMode.wpmodexToolbarLG,
 // EditorGUI.wpguiGutter +
 EditorGUI.wpguiHorzScrollBar |
 EditorGUI.wpguiPanelH1|
 // EditorGUI.pguiPanelH2 |
 EditorGUI.wpguiPanelV1|
 EditorGUI.wpguiPanelV2 |
 EditorGUI.wpguiRuler  |
 EditorGUI.wpguiVertRuler  |
 EditorGUI.wpguiVertScrollBar
 , EditorGUI.wpguiDontSet);
```

Parameters

Mode

a) wpmodSingleEditor = 0: Simple editor

b) wpmodDoubleEditor = 1: two Editors - working with 2 different texts. Both text are using the same
This mode is reserved for the future to easy the creation of reports. In one editor the report template
the other displays the created report. Usually the DLL works with the upper, Editor 1. To switch to Edi
SelectEditor can be used. Most of the events pass a parameter 'editor' which is 1 for the upper, 2 for t

c) wpmodSplitEditor = 2: Editor with split screen - the same text is edited in both windows. But each
different zoom level and layout mode. Both windows can show a different position in the text.Most of t
parameter 'editor' which is 1 for the upper, 2 for the lower editor.

d) wpmodSplitThumbnails = 3: Editor and Preview/Thumbnail display. This mode is almost identical to
mode. The lower editor is readonly though.

In addition to the general operation modes certain special features can be activated using property Mo
flags are possible:

XMode

```
wpxtraToolbar    = 1; - Select the 16x16 toolbar
wpxtraToolbarLG    = 2 - Select 24x24 Toolbar
wpxtraSpellcheck = 4 -  Activate the Spellcheck
wpxtraTables     = 8 -  Activate support for tables (in toolbar)
wpxtraPDFExport  = 16 -  Activate the wPDF PDF Export
wpxtraReporting  =32 -  Activate the WPReporter in EditorA (cannot be used in 'Split'
wpxtraPremium = 64 - Activate the premium features (support for footnotes, columns an
```

Please note that setting a flag will have no effect if you do not have the license to use i.e. the PDF exp
features.

This parameter selects the graphical user interface (gui) elements used by the editor 1.

Gui1

```
  wpguiRuler       = 2, // Select a ruler
  wpguiVertScrollBar=4, // Select the vertical scrollbar
  wpguiHorzScrollBar=8, // Select the horizontal Scrollbar
  wpguiVertRuler   =16, // Select Vertical Ruler
  wpguiGutter      =32, // Select Gutter (pagno)
  wpguiPanelV1     =64, // Select the panel in top right corner
  wpguiPanelV2     =128,// Select the panel in the bottom rigth corner
  wpguiPanelH1     =256,// Select the panel in the bottom left corner
  wpguiPanelH2     =512; //Panel on the right side of wpguiPanelH1
  wpguiUnitsAreInch=1024;//Units used for ruler are inch instead of cm
```

If the flag wpguiDontSet (1) was used this parameter will be ignored. This is useful if you call SetEdito
modify an editor.

Gui2 This parameter selects the graphical user interface (gui) elements used by the editor 2. If you do not
editor simply specify wpguiDontSet.

Category
Modify the layout of the text display 153

8.1.2.23 SetEnableFlags

Applies to
WPDLLInt 94

Declaration
```
procedure SetEnableFlags(Editor: Integer; Flags: Integer);
```

Description
Using the method SetEnableFlags it is possible to modify the editor #1 or #2:
Certain options which are not active by default can be activated by setting one bit in the parameter:

---

bit 1: wpoptDropImgCreatesLinkedImage=1 - when an image file is dragged onto the editor a link to the file is created. (only the path name is stored).

bit 2: wpoptDropImgCreatesMovableParObject=2 - when an image file is dragged onto the editor a movable image will be created - it is positioned relatively to a paragraph.

bit 3: wpoptDropImgCreatesMovablePageObject=4 - when an image file is dragged onto the editor a movable image will be created - it is positioned relatively to a page.

bit 4: wpoptDropImgCreatesNoWrapImage=8 - when an image file is dragged onto the editor a movable image will be created with text wrapping switched off.

bit 5: **wpoptFormularMode=16** - Activates the formular mode. The user can only edit certain mail merge fields which are marked to be editable (Mode 398=2).

bit 6: wpoptAllowCreateTableInTable=32 - Tables may be created within other tables.

bit 7: wpoptShowSpecialChars=64 - Special characters, such as CR, NL and FF are displayed.

Please also see method SetBProp 197 which allows it to toggle more than 100 internal proerty flags to adjust the editor handling to you needs.

8.1.2.24 SetLanguage

Applies to
WPDLLInt 94

Declaration
```
function SetLanguage(const LanguageID: WideString): WordBool;
```

Description
This property changes the language of the GUI. The parameter is the identifier of the language, for example "DE" for german.

The PCC file contains the extensible language definition in form of XML data. Please see the chapter about packages files 424.

8.1.2.25 SetLayout

Applies to
WPDLLInt 94

Declaration
```
procedure SetLayout(const LayoutFile: WideString; const LayoutName: WideString;
const LayoutPASS: WideString; const MainXML: WideString; const SecondXML:
WideString);
```

Description
The method SetLayout() is used to load the description file for the user interface. The editor cannot display a toolbar without such a file.
If you specify a relative path the directory which contains the engine DLL will be used as basis.

Example:
```
SetLayout(".\\buttons.pcc", "default", "", "main", "main");
```

The .NET assembly also implements the SetLayout method with only two parameters, LayoutFile and LayoutPW. The other parameters will use the values "default", "main", "main".

So you can also type:
```
SetLayout(".\\buttons.pcc","");
```

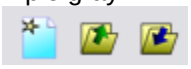
To select from different design styles use function Command(ID,Mode,0) 107 with ID **WPDLL_COM_ToolbarDesign** (5902) and a number between 1 and 4:

1 = Titanium (default)

2 = Brushed Metal - use Command(9501, 1) to make the dialogs use this background

3 = Shaded. Use Command 9503 and 9504 to change colors)

4 = Simple gray



The shading colors can also be changed, see "Command 107".

Parameters

LayoutFile	the path to the layout file, i.e. "Buttons.PCC".
LayoutName	the name of the layout entry in the package file, i.e. "default".
LayoutPW	the password used for the PCC file or an empty string.
LayoutMainXML	the XML branch for the main editor, i.e. "main".
LayoutSecondXML	the XML branch for the second editor (can be the same as LayoutMainXML).

8.1.2.26 SetLayoutMode

Applies to
WPDLLInt 94

Declaration
```
procedure SetLayoutMode(LayoutMode: TxWPLayoutMode; AutoZoom: TxWPAutoZoom;
ZoomValue: Integer);
```

Description
This method selects the layoutmode and zooming used by editor #1.

Parameters

Layout	0 = Normal 1 = Word wrap - format to window margins 2 = Show Manual Page Breaks 3 = Page Gap = like "Normal" but show gray area between pages 4 = Extended PageGap = also show left+right margins 5 = Shrunken Layout = hide header + footer area 6 = Layout = full page but no header/footer 7 = Full Layout = full page + header/footer 8 = Dual PageView = 2 * full page 9 = ThumbNail View
AutoZoom	0=Off 1=Page Width 2=Full Page 3=AdjustColumnCount 4=Show as many pages as possible in one row.
Zooming	Only with AutoZoom=Off - set zooming in percent.

8.1.2.27 SetLayoutMode2

Declaration
```
procedure SetLayoutMode2(LayoutMode: TxWPLayoutMode; AutoZoom: TxWPAutoZoom;
ZoomValue: Integer);
```

Description
Works like SetLayoutMode 117 but for editor #2.

8.1.2.28 SetSplitterPos

Applies to
WPDLLInt 94

Declaration
```
procedure SetSplitterPos(Percentage: Integer);
```

Description
This method sets - if two editors are visible - the splitter position in per cent.

Tip: It is possible to hide an editor temporarily using Memo.Hidden=true 172.

8.1.2.29 SetText

Applies to
WPDLLInt 94

Declaration
```
procedure SetText(Editor: Integer; const Data: WideString; const Format:
WideString; InsertText: WordBool);
```

Description

This methods assigns or inserts text suing a given format in editor one or two.

Category
Load and Save 150

8.1.2.30 SetTextVar / SetBytes

Applies to
WPDLLInt 94

Declaration:
```
bool SetTextVar(int Editor,object Data, string Format, bool InsertText);
```

The .NET Assembly also defines:
```
public bool SetBytes(int Editor, byte[] Data, string Format, bool InsertText)
```

Description
Both methods can be used to load the text from a variant / array of bytes. This has the advantage that RTF data (which is plain ASCII code) does not have to be passed as unicode string.

Parameters

Editor	The editor to update: 1 or 2
Data	THe text as OleVariant. If it is null nothing happens, if it is empty the text will be cleared. Please use an array of bytes to load data from a data base blob.
Format	A string which specifies the loading options.
InsertText	true to insert text, false to replace all text with this data

Returns
True if loading went ok.

Also see GetTextVar 111!

8.1.2.31 SpecialTextAttr

Applies to
WPDLLInt 94

Declaration
```
function SpecialTextAttr(Select: TxSpecialTextSel): IWPCharacterAttr 292;
```

Description
This method makes it possible to modify the appearance of hyperlinks, fields and other text. Select may have the following values:

```
public enum SpecialTextSel // used with IWPCharacterAttr
{
 wpHiddenText = 0,
 wpFootnote = 1,
 wpInsertpoints = 2,
 wpHyperlink = 3,
 wpSPANStyle = 4,
 wpAutomaticText = 5,
 wpProtectedText = 6,
 wpBookmarkedText = 7,
 wpInsertedText = 8, // reserved
 wpDeletedText = 9,  // reserved
 wpWordHighlight = 10,// reserved
 wpFieldTextObjects = 11
```

```
}
```

Example:

```
WPDLLInt1.SpecialTextAttr(
    SpecialTextSel.wpInsertpoints).Hidden = true;
WPDLLInt1.SpecialTextAttr(
    SpecialTextSel.wpInsertpoints).CodeTextColor = 0;
```

Category
Hyperlinks and Bookmarks 149
Mailmerge 154

8.1.2.32 wpaCheck

This method is used to check the selected/hidden/disabled state for the action identified by the provided name.

Applies to
WPDLLInt 94

Declaration
```
function wpaCheck(const WPA: WideString): Integer;
```

Description
The return value is a bitfield:
bit 1: action is enabled
bit 2: action is selected (menu shows check, button is pressed)
bit 3: action is hidden, it is not available
bit 7: This bit is always set

Category: Action Names 417

8.1.2.33 wpaExec

Execute an action identified by an ID.

Applies to
WPDLLInt 94

Declaration
```
function wpaExec(wpaID: Integer; const param: WideString): WordBool;
```

Description
The additional string parameter show be usually empty. Only to change the font, size or color provide the parameter as string.

Category: Action Names 417

8.1.2.34 wpaGetFlags

This method retrieves the state of all WPA actions.

Applies to
WPDLLInt 94

Overloaded Variants

Procedure wpaGetFlags(EditoNumber: Integer); 120
Function wpaGetFlags(Editor: Integer): WideString; 120

Declaration
```
procedure wpaGetFlags(Editor: Integer): WideString;
```

Description
This method is used witin the event OnUpdateGUI 143 to retrieve the current flags in form of an array. The OCX and the .NET assembly behave a bit differently:

a) OCX
The wpaGetFlags implementation returns a string with each character contains the state bits for one action.

```
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
s = WPDLLInt1.wpaGetFlags(0) ' this are the flags for the current editor
i = WPDLLInt1.wpaGetID("bold") ' this is the id for the command to toggle 'bold'
Bytes = StrConv(s, vbFromUnicode) ' convert string to bytes
If Bytes(i) And 2 Then BoldMenu.Checked = True Else  BoldMenu.Checked = False
```

b) .NET
The function wpaGetFlags(EditorNr : Integer) fills a byte array. This array can be used easily and no format conversion is required.

```
private void wpdllInt1_OnUpdateGUI(object Sender, int Editor, int UpdateFlags, int StateF
{
 int wpa_bold = wpdllInt1.wpaGetID("Bold");
 byte[] stateflags = wpdllInt1.wpaGetFlags(0); // Current editor
 BoldMenu.Pushed = (stateflags[wpa_bold] & 2)!=0;
}
```

Parameters	
EditorNr	The number of the editor to examine, 1 or 2 Use 0 to check the active editor.

Returns
An array with a bitfield for each element (Byte or Char):
bit 1: action is enabled
bit 2: action is selected (menu shows check, button is pressed)
bit 3: action is hidden, it is not available
bit 7: This bit is always set to avoid #0 entries.

Category: Action Names 417

8.1.2.35 wpaGetID

This method is used to get the wpa ID for a given WPA action name.

Applies to
WPDLLInt 94

Declaration
```
function wpaGetID(const wpaName: WideString): Integer;
```

Description
Please check out this chapter in the manual (PDF):
"Tasks/Getting Started/First C# Application"
It describes how to create a menu from an array of action names. A special utility class "WPAMenuItem" is used for automatic linking to the editor.
A similar approach can be used to use an external tool bar.

Category: Action Names 417

8.1.2.36 wpaProcess

Applies to
WPDLLInt 94

Declaration
```
procedure wpaProcess(const wpaName: WideString; const param: WideString);
```

Description
This method executes a WPA action. A parameter can be specified - this is only used by a few actions.

Category: Action Names 417

8.1.2.37 wpaSetFlags

This method can be used to update the flags provided by wpaGetFlags.

Applies to
WPDLLInt 94

Declaration
```
procedure wpaSetFlags(Editor: Integer; Start: Integer; Count: Integer; const
AllFlags: WideString);

public void wpaSetFlags(int Editor, int Start, int Count, byte[] AllFlags)
```

Description
It can be only used within the OnUpdateGUI event. With it you can manipulate the states of the buttons in the editor i.e. hide or disable some. You can use wpaGetFlags to first get the states of all buttons and the use wpaSetFlags to change the flags, or you use a new bytes array (with OCX use a string) only for the action you need to update.
Note: If the "start" parameter is >0, the first element of the passed array will be mapped to the flag property of the action with the id "start".

Example: Hide the "new" button:

```
private int wpaNew = -1;
private void wpdllInt1_OnUpdateGUI(object Sender, int Editor, int UpdateFlags, int StateF
{
 byte[] states;
 states = wpdllInt1.wpaGetFlags(Editor);
 // Store
 if (wpaNew<0) wpaNew = wpdllInt1.wpaGetID("New");
 states[wpaNew] |= 4; // bit 3 - hide it
 wpdllInt1.wpaSetFlags(Editor,0,states.Length, states);
}
```

This example uses the start/count parameter to only change one button.

```
private int wpaNew = -1;
private void wpdllInt1_OnUpdateGUI(object Sender, int Editor, int UpdateFlags, int StateF
{
 byte[] states = new byte[1];
 if (wpaNew<0) wpaNew = wpdllInt1.wpaGetID("New");
 states[0] = 4; // bit 3 - hide it
 wpdllInt1.wpaSetFlags(Editor,wpaNew,1, states);
}
```

The same as VB6 code:

```
Option Explicit
Dim wapNew As Integer
Private Sub WPDLLInt1_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)
  If wapNew = 0 Then
     wapNew = WPDLLInt1.wpaGetID("New")
  End If
  WPDLLInt1.wpaSetFlags 1, wapNew, 1, "E"
End Sub
```

Category: Action Names 417

8.1.2.38 ReleaseInt

This method (it is exclusive to the .NET assembly) releases a reference to some of the 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.

Background: Some interfaces modify live objects inside the text. This can be paragraph of number styles, text layers or image objects. Those objects can be freed while the text is edited. So it is important to release any connection to the object as soon as possible, this is what ReleaseInt does.

It is required to call ReleaseInt for the interfaces:

IWPDataBlock 296, retrieved by Memo.BlockAdd 176, BlockAppend 177 and BlockFind 178
IWPNumberStyle 328, retrieved by Memo.GetNumberStyle 183
IWPTextObj 396, retrieved by TextCursor.CPOpenObj 231
IWPParInterface 339, retrieved by IWPDataBlock.CurrPar 300
IWPAttrInterface 276, retrieved by IWPDataBlock.CurrParAttr 300

It is not required to call it for IWPEditor, IWPMemo, IWPTextCursor and any interfaces which are passed as event parameters since they are automatically. released. It is also not required for CurrAttr, TextAttr, CurSelAttr etc. The new .NET dlls automatically make sure such buffered interfaces are not freed.

8.1.2.39 SetDLLName

This method is exclusive to the .NET assembly. **Please use it as early as possible in the program:**

```
// look in same directory as assembly
Assembly TextDynamicAssembly = Assembly.GetAssembly(typeof(WPDLLInt));
string TextDynamicDLL = Path.GetDirectoryName(TextDynamicAssembly.Location) + "\\WPTextDL
// If not existent there look in same directory as executable
if (!File.Exists(TextDynamicDLL))
{
   TextDynamicDLL = Path.GetDirectoryName(Application.ExecutablePath) + "\\WPTextDLL01.dl
}
// Set the DLL name (prelaod the DLL)
WPDynamic.WPDLLInt.SetDLLName(TextDynamicDLL);
```

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.

8.1.2.40 DrawToBitmap

```
public void DrawToBitmap(Bitmap bmp, Rectangle Rect)
```

This method (only defined in .NET wrapper) makes a screenshot of the complete editor and the toolbars at the position Rect.Left and Rect.Top on the provided bitmap. Please note that the scrollbars will not be visible if the editor is hidden.

Note: Rect.Width and Rect.Height are currently not used.

Example:

```
Bitmap ScreenShot = new Bitmap(wpdllInt1.Width, wpdllInt1.Height);
wpdllInt1.DrawToBitmap(ScreenShot, new Rectangle(0, 0, wpdllInt1.Width, wpdllInt1.Height)
ScreenShot.Save(@"c:\printed.bmp");
```

8.2 Events

8.2.1 RTF2PDF / TextDynamic

8.2.1.1 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 124 to temporarily update the properties of the object.

Category
Image Support 149

8.2.1.2 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 409 let you examine the current writing mode and path.
TextObject	The interface IWPTextObj 396 gives you access to the object properties.
DontSave	If this variable is changed to "true" the object will not be saved.

Category
Image Support 149

8.2.1.3 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.

8.2.1.4 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 224 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 224. 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 224 call InputParagraph 244 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 339 makes it possible to modify the text in this cell.
CellAttr	The IWPAttrInterface 276 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 148
Table Support 158

8.2.1.5 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 180. 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 296 is passed as parameter. The parameter EventParam is the integere value which was provided to EnumDataBlocks().

Category
Callback Functions 148

8.2.1.6 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 180, EnumParSiblings 181, EnumSelParagraphs 181 and EnumParStyles 181. It is also used by WordEnum 264.

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** 108 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) 107. (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, int Count, WPDynamic.IWP
 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 139 can be used.

Category
Callback Functions 148

8.2.1.7 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 148

8.2.1.8 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()

8.2.1.9 OnFieldGetText

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 189) and the reporting engine (see interface IWPReport 372).
It is used to fill fields with data during the process of mail merge or report creation. The mail merge is started with MergeText 189. Inside the merge process the event OnFieldGetText is triggered for all fields.

The provided interface Contents 307 makes it easy to change the text displayed by the field.

```
Private Sub WPDLLInt1_OnFieldGetText(ByVal Editor As Long, ByVal FieldName As String, ByV
  If FieldName='NAME' then Contents.StringValue = 'Julian Ziersch'
End Sub
```

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 314 and LoadPicture 314. 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 313. You can also load a file with formatted text using LoadText 314

To create a table with data inside the field execute AddTable 310. The event OnCreateNewCell 125 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 312. The field markers, not the contents will be removed.

An interface to access the current field is provided as property FieldObject 313. You can use this interface to read other properties of the merge field.

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 EventField 101 instead.

Note
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!

8.2.1.10 OnGetSpecialText

Member of WPDLLInt 94
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 298.

Category
Hader and Footer Support 148

8.2.1.11 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
```

8.2.1.12 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 396.

8.2.1.13 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.

8.2.1.14 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.

8.2.1.15 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 326. The modified size will be only used for the page with the number "PageNr".

Category
Modify the layout of the text display 153

8.2.1.16 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.

8.2.1.17 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.

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-YMargin*2);

   // 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);
}
```

8.2.1.18 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 372).

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 94

8.2.1.19 OnReportState

This event is use by the reporting engine (see interface IWPReport 372). (The product RTF2PDF does not yet support reporting!)

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.

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

```
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;
}
```

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.

Parameters

Name	This is the name 378 of the band.
State	The parameter state 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 377 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 129 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.

Category
Reporting 94

8.2.1.20 OnTextObjectGetText

Declaration C#
OnObjectGetText(**Object** Sender, int Editor, **int** PageNr, IWPTextObj 396 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 399 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, B
  Obj.Params = "<undefined:" + Obj.Name + ">"
End Sub
```

8.2.2 Exclusive to TextDynamic

8.2.2.1 OnBeforeOverwriteFile

Member of WPDLLInt 94
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 150

8.2.2.2 OnButtonClick

Member of WPDLLInt 94
Declaration C#
OnButtonClick(Object Sender, int Editor, IWPDllButton Def)

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

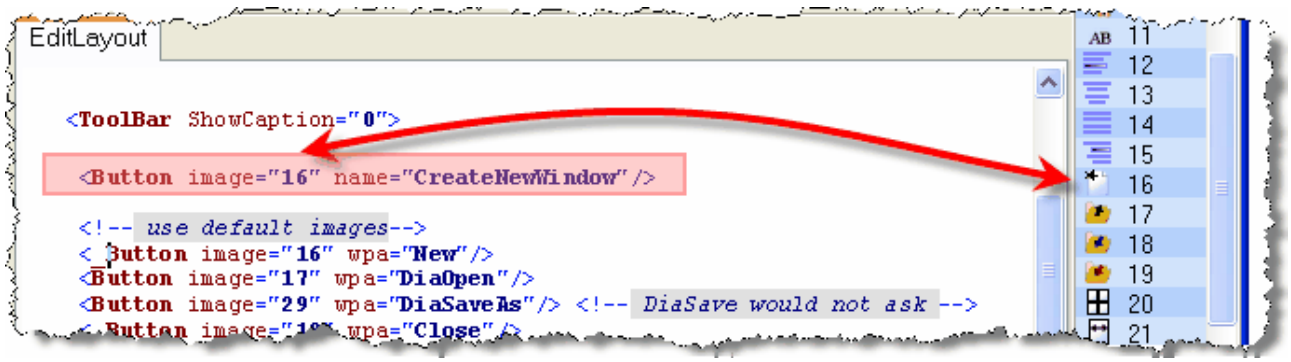
This event is triggered when the user clicks on a custom button or menu.

The interface IWPDllButton 303 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.IWPDllButton 303)
   If Def.Name 305 = "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 the property EventButton 101 instead.

8.2.2.3 OnChangePosition

Member of WPDLLInt 94
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 104 and CurrMemo.TextCursor 168.

However, in general we recommend to use OnUpdateGUI 143 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 157

8.2.2.4 OnChangeSelection

Member of WPDLLInt 94
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 157

8.2.2.5 OnChangeText

Member of WPDLLInt 94
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 137 will be triggered before the change.

8.2.2.6 OnChangeViewMode

Member of WPDLLInt 94
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 172 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 157

8.2.2.7 OnChangingText

This event is triggered before the text is modified.

Member of WPDLLInt 94
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.

8.2.2.8 OnClick

Member of WPDLLInt 94
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.

8.2.2.9 OnClickCreateHeaderFooter

Member of WPDLLInt 94
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 148

8.2.2.10 OnClickPage

Member of WPDLLInt 94
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.

8.2.2.11 OnDblClick

Member of WPDLLInt 94
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.

8.2.2.12 OnFieldEnter

Member of WPDLLInt 94
Declaration C#
OnEnterField(**Object** Sender, **int** Editor, **string** Fieldname, IWPTextObj 396 Field)

Declaration OCX
OnFieldEnter(ByVal Editor As Long, ByVal FieldName As String, ByVal Field As WPTDynInt. IWPTextObj)

When in formular mode (see SetEnableFlags 116) this event will be triggered when the cursor was moved into a different field.

8.2.2.13 OnFieldLeave

Member of WPDLLInt 94
Declaration C#
OnLeaveField(**Object** Sender, **int** Editor, **string** Fieldname, IWPTextObj 396 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 116) 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.

8.2.2.14 OnHyperlink

Member of WPDLLInt 94
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 396 you can also examine and modify the hyperlink: TextObj.EmbeddedText is the visible text, TextObj.Command is the URL.

Category
Hyperlinks and Bookmarks 149

8.2.2.15 OnMouseDown

Member of WPDLLInt 94
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.

8.2.2.16 OnMouseDownWord

Member of WPDLLInt 94
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 339 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) 108.

If you use this event to show a custom popup dialog please use Memo.SetBProp(0, 19, -1) 198 to deactivate the default popup dialog.

Tip: If custom spellchecking was activated using Command(907) 107 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;
}
```

8.2.2.17 OnMouseMove

Member of WPDLLInt 94
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.

8.2.2.18 OnMouseUp

Member of WPDLLInt 94
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:

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 Long, ByVal X As Long
    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
```

8.2.2.19 OnShowHint

Member of WPDLLInt 94
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 Ignore)
{
   stHint.Text = Hint;
   Ignore = True;
}
```

8.2.2.20 OnTextObjectMouse

Member of WPDLLInt 94
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 396 you can manipulate it. To switch a text frame (TextDynamic "Premium") you can call procedure Contents_edit 402.

8.2.2.21 OnUndoChange

Member of WPDLLInt 94
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 157

8.2.2.22 OnUpdateGUI

This is the main event to update any custom toolbar and menu.

Member of WPDLLInt 94
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 120 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)

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

StateFlags	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. 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

Category
Display Status Information 157

8.2.2.23 OnKeyDown

Member of WPDLLInt 94
Declaration C#
KeyDown(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
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.

8.2.2.24 OnKeyPress

Member of WPDLLInt 94
Declaration C#
KeyPress(KeyPressEventArgs e)
Here the standard event is used. You can cast e to wpKeyPressEventArgs which contains the property Editor.

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) 144**:**

```
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
```

8.2.2.25 OnKeyUp

Member of WPDLLInt 94
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

8.2.2.26 OnLeaveEditor

Member of WPDLLInt 94
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)

8.2.2.27 OnCompleteWord

Member of WPDLLInt 94
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 168. You can also use LoadFromString 188 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
```

8.2.2.28 OnEnterEditor

Member of WPDLLInt 94
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.

8.3 Categories

8.3.1 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	Methods
WPDLLInt.AttrHelper 97	IWPFieldContents.FieldAttr 313

IWPAttrInterface.CharAttrIndex 279

IWPParInterface.CharAttr 352
IWPParInterface.GetCharAttr 354
IWPParInterface.ReplaceCharAttr 364
IWPParInterface.SetCharAttr 365
IWPTextCursor.WordCharAttr 263

8.3.2 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 283
GetFontSize 283

Write Property

SetFontface 288
SetFontSize 289

Other Methods

IWPAttrInterface.ExcludeStyles 281
IWPAttrInterface.GetStyles 284
IWPAttrInterface.IncludeStyles 286
IWPAttrInterface.SetStyles 289
IWPAttrInterface.ToggleStyle 291

8.3.3 Coordinate Conversion Category

Methods to retrieve and convert coordinates used inside the editor

Description
Since the TextDynamic control can host one toolbar, two editors and several tool panels the coordinates used by each of the editor are notr always measured relatively to the upper left corner of the control. So we provided methods to calculate position coordinates.

Properties
none

Methods

IWPMemo.ClientToScreen 213
IWPMemo.GetMouseXY 213
IWPMemo.GetXY 186
IWPMemo.ScreenToClient 216

8.3.4 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
none

Methods
IWPMemo.GetRTFVariable 186
IWPMemo.SetRTFVariable 207

8.3.5 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
OnCreateNewCell 125
OnEnumDataBlocks 126
OnEnumParOrStyle 127
OnEnumTextObj 127

Methods
IWPFieldContents.AddTable 310
IWPMemo.EnumDataBlocks 180
IWPMemo.EnumParagraphs 180
IWPMemo.EnumParSiblings 181
IWPMemo.EnumParStyles 181
IWPMemo.EnumSelParagraphs 181
IWPMemo.EnumTextObj 182
IWPTextCursor.AddTable 224

8.3.6 Hader 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".

See Properties of
IWPDataBlock 296

Events

Methods
IWPMemo.BlockAdd 176
IWPMemo.BlockAppend 177

OnClickCreateHeaderFooter 137
(not in RTF2PDF)
OnGetSpecialText 129

IWPMemo.BlockFind 178
IWPMemo.FindFooter 182
IWPMemo.FindHeader 182
IWPTextCursor.GotoBody 234
IWPTextCursor.InputFooter 239
IWPTextCursor.InputHeader 240

8.3.7 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
OnHyperlink 139

Methods
Memo.SpecialTextAttr 169
IWPFieldContents.InputHyperlink 313
IWPMemo.GetObjAtXY 183
IWPTextCursor.InputBookmark 235
IWPTextCursor.InputHyperlink 241
IWPTextCursor.MoveToBookmark 255

Tip: You can use IWPMemo.EnumTextObj 182 to read the url and text of all hyperlinks.

8.3.8 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 396 interface is used to read and write the properties of the reference object.

Events
OnAfterSaveImage 124
OnBeforeSaveImage 124

Interface to the object placeholder
IWPTextObj 396

Methods
Memo.InsertGraphicDialog 214
IWPParInterface.InsertNewObject 357
IWPTextCursor.InputImage 241
IWPTextCursor.InputPicture 244
IWPTextObj.LoadFromFile 405
IWPTextObj.SetContentsID 407

8.3.9 Load and Save Category

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.

Properties

WPDLLInt.Text 103
WPDLLInt.Text2 104
WPDLLInt.TextFormat 104

IWPDataBlock.Text 299
IWPMemo.LastFileName 167
IWPMemo.Modified 173
IWPMemo.SelText 168
IWPMemo.Text 168

Methods

WPDLLInt.GetText 111
WPDLLInt.GetTextVar 111
WPDLLInt.SetText 118
WPDLLInt.SetTextVar 119

IWPFieldContents.LoadText 314
IWPMemo.GetPageAsMetafile 184
IWPMemo.InsertGraphicDialog 214
IWPMemo.Load 215
IWPMemo.LoadEx 215
IWPMemo.LoadFromFile 187
IWPMemo.LoadFromStream 188
IWPMemo.LoadFromString 188
IWPMemo.LoadFromVar 188

IWPMemo.Save 215
IWPMemo.SaveAs 215
IWPMemo.SavePageAsMetafile 193
IWPMemo.SaveToFile 194
IWPMemo.SaveToStream 194
IWPMemo.SaveToString 195
IWPMemo.SaveToVar 196
IWPParInterface.LoadFromFile 359
IWPParInterface.LoadFromString 359
IWPParInterface.SaveToString 364
IWPPdfCreator.Print 275
IWPTextObj.LoadFromFile 405
IWPTextObj.LoadFromStream 405

When you work with a **data bound editor** you need to use TextCursor.CheckState 225(10) to trigger the PropChanged event at first chance before any code which updates the text.

Please also see chapter "HTML/E-MAIL loading and saving 79" in the manual.

If you want to display a **split editor** with the HTML source (HTML code) and formatted text you can use TextCommandStr 209(5, ..) .

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 296" 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.

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							
ignorerowmerge	If this flag is set the rowmerge property will be ignored when reading RTF	X								

	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 (default = "\" which is the current directory)				X					
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				X					

	the information of the default attributes of the document									
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

8.3.10 Modify the layout of the text display Category

Properties and Methods which change the way the text is displayed

Description
This category contains links to commands to update the formatting on screen. Only the property PageSize will make persistent changes to the text. The other properties only change the text virtually.

Properties
IWPMemo.AutoZoom 169
IWPMemo.ColorDesktop 169
IWPMemo.ColorHighlight 169
IWPMemo.ColorHighlightText 170
IWPMemo.ColorPaper 170
IWPMemo.CurrentZooming 163
IWPMemo.FormCompletion 171
IWPMemo.GUIMode 171
IWPMemo.LayoutMode 172
IWPMemo.PageSize 167
IWPMemo.PageSizeList 167
IWPMemo.ShowFields 174
IWPMemo.TopOffset 175
IWPMemo.WordWrap 175
IWPMemo.Zooming 175

Methods
WPDLLInt.SetEditorMode 114

Event
OnMeasurePage 131

8.3.11 Logical MDI Support Category

Manage multiple documents in one control.

Description
TextDynamic has the ability to store several documents in one editor. If you create a "double editor" you can work with 2 texts in two editor windows. But this is not what is meant here. With this methods you can create many documents and hold them all in the editor. You can switch between them and remove the documents if required. This all works like an MDI application, only that just one editor is used.

Properties
none

Methods
IWPMemo.RTFDataAdd 191
IWPMemo.RTFDataAppendTo 191

IWPMemo.RTFDataDelete 192
IWPMemo.RTFDataSelect 193

8.3.12 Mailmerge Category

Update and read out data fields in the text

Description
The mailmerge process offered by TextDynamic is one of its most popular features. It can replace the contents of merge fields with formatted text or read out the text in case the user has edited it. This works since the mail merge is not destructive, the field will not be replaced like a Search&Replace would do, but only the data within the field is. 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 237 and CurrObj.Mode 398=2 you can create an "Edit Field" which can be changed in form completion 171 mode while the rest of the text is protected.

Properties

IWPFieldContents.StringValue 309
IWPMemo.LabelDef 166
IWPMemo.ShowFields 174
IWPMemo.CurrObj 163 this is the field which was added using InputField 237 last.

The **event** OnFieldGetText 129 receives a reference to IWPFieldContents 307 which contains many properties.

Methods

IWPFieldContents.ContinueOptions 311
IWPMemo.AppendOtherText 175
IWPMemo.DeleteParWithCondition 179
IWPMemo.MergeText 189
IWPMemo.RTFDataAppendTo 191
IWPTextCursor.FieldsFromTokens 233 (=ReplaceTokens)
IWPTextCursor.InputField 237

Note: The field markers << and >> are usually visible. To hide the markers use
`Memo.SpecialTextAttr(wpInsertpoints).Hidden = True`

8.3.13 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 222".

Properties
none

Methods

IWPTextCursor.MarkerCollect 253
IWPTextCursor.MarkerCollectAll 254
IWPTextCursor.MarkerCPPosition 254

IWPTextCursor.MarkerDrop 254
IWPTextCursor.MarkerGoto 254
IWPTextCursor.MarkerSelect 255

If you just need to move back or forward in the text please also do not use "CPPosition 222" but one of these methods from IWPTextCursor 219:

CPMoveAfterTable 227
CPMoveBack 227
CPMoveBeforeTable 227
CPMoveNext 228
CPMoveNextBand 228
CPMoveNextCell 228
CPMoveNextGroup 228
CPMoveNextObject 228
CPMoveNextPar 229
CPMoveNextRow 229
CPMoveNextTable 230
CPMoveParentTable 230
CPMovePrevCell 230
CPMovePrevObject 230
CPMovePrevPar 230
CPMovePrevRow 231
CPMovePrevTable 231

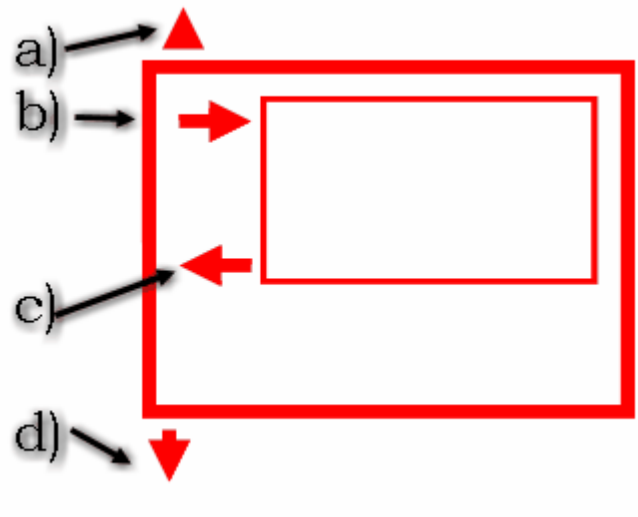
8.3.14 Lowlevel Paragraph IDs Category

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 355
Set ID:
IWPParInterface.SetPtr 366

a) previous in list (i.e. previous paragraph, cell to the left)
IWPParInterface.GetPtrPrev 356

b) first child or 0 (i.e. first cell in table row, first row in table, second paragraph in cell)
IWPParInterface.GetPtrChild 355

c) parent or 0 (i.e parent row of cell, parent table of row)
IWPParInterface.GetPtrParent 356

d) next in list (i.e. next paragraph, cell to the right)
IWPParInterface.GetPtrNext 356

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 368 - they provide a save way to loop all

paragraphs in the current document or text block.

8.3.15 Printing Category

Properties and methods for printing

Properties
IWPMemo.PageSizeList 167
IWPMemo.PrintParameter 167

Methods
Select Printer
Memo.TextCommandStr(10, name). 212
or WPDLLInt.SelectPrinter 114

Start Printing:
Memo.TextCommandStr(11, range). 212
or **IWPMemo.Print** 190 and
IWPMemo.PrintPages 190

Create printing cue:
WPDLLInt.BeginPrint 106
WPDLLInt.EndPrint 106

Alternatively you can use
Memo.TextCommandStr_13 212
and Memo.TextCommandStr_ID_14 212

to use the optional PDF export:
IWPPdfCreator.Print 275

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, param = title
WPCOM_END_PRINT = 1324;       // use beginprint/endprint
```

8.3.16 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.

Note: TextDynamic works with an unlimited count of undo steps.

Properties

Methods
IWPMemo.CopyToClipboard 178
IWPMemo.CutToClipboard 178
IWPMemo.PasteFromClipboard 190
IWPTextCursor.EnabledProtection 232
IWPTextCursor.SelectAll 259
IWPTextCursor.SelectLine 259
IWPTextCursor.SelectParagraph 259
IWPTextCursor.SelectTable 260

IWPTextCursor.Undo 263

IWPTextCursor.UndoClear 263
IWPTextCursor.Redo 257
IWPTextCursor.DisableUndo 232
IWPTextCursor.EnableUndo 232

8.3.17 Display Status Information Category

Show current page, page count, zooming

Description
TextDynamic supports several events to update a status bar. The most powerful is OnUpdateGUI which also has the advantage that it is triggered when the application is idle.

Properties
IWPMemo.CurrentZooming 163
IWPTextCursor.CPLineNr 220
IWPTextCursor.CPPageNr 221
IWPTextCursor.CPPosInLine 221
IWPTextCursor.CPPosition 222
IWPTextCursor.PageCount 223

Events
WPDLLInt.OnChangePosition 135
WPDLLInt.OnChangeSelection 136
WPDLLInt.OnChangeViewMode 136
WPDLLInt.OnUndoChange 142
WPDLLInt.OnUpdateGUI 143

Method
IWPMemo.Statistic 207

8.3.18 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 165
IWPMemo.CurrStyleAttr 165
IWPParInterface.StyleName 348

Methods
IWPMemo.DeleteStyle 180
IWPMemo.EnumParStyles 181
IWPMemo.SelectStyle 196

IWPTextCursor.CPStylePtr 222
IWPTextObj.StyleName 400

IWPTextCursor.InputParagraph 244

8.3.19 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 224 and use the event OnCreateNewCell 125 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 268 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 248. You can also move the cursor using CPMoveNextRow 229, 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 249, 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 339 to make it a table object: par.SetParType 366((int)ParagraphType.Table). Now you can use AppendChild 350 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 368 methods, or you save the paragraph ID and use par.SetPtr 366(id).

Please see the example code in topic IWPDataBlock.AppendParagraph 301.

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 342
IWPParInterface.CellName 342
IWPParInterface.IsColMerge 343
IWPParInterface.IsFooterRow 343
IWPParInterface.IsHeaderRow 344
IWPParInterface.IsRowMerge 345
IWPParInterface.WidthTW 349
IWPTextCursor.CPTableColNr 223
IWPTextCursor.CPTablePtr 223
IWPTextCursor.CPTableRowNr 223

IWPDataBlock.AppendParagraph 301
IWPFieldContents.AddTable 310
IWPMemo.Tables_WidthFixed 208
IWPMemo.Tables_WidthPC 208
IWPParInterface.AppendChild 350
IWPParInterface.SetParType 366
IWPReportBand.AddTable 381
IWPTextCursor.AddTable 224
IWPTextCursor.AppendRow 225
IWPTextCursor.CombineCellHorz 226
IWPTextCursor.CombineCellVert 227
IWPTextCursor.CPMoveAfterTable 227
IWPTextCursor.CPMoveBeforeTable 227

IWPTextCursor.CPMoveNextCell 228
IWPTextCursor.CPMoveNextRow 229
IWPTextCursor.CPMoveNextTable 230
IWPTextCursor.CPMoveParentTable 230
IWPTextCursor.InputCell 235
IWPTextCursor.InputRowEnd 245
IWPTextCursor.InputRowStart 246
IWPTextCursor.InputTable 249
IWPTextCursor.SelectTable 260
IWPTextCursor.SelectTableColumn 260
IWPTextCursor.SelectTableRow 260
IWPTextCursor.SetTableLeftRight 262
IWPTextCursor.TableClear 262
IWPTextCursor.TableDelete 263

Event
OnCreateNewCell 125

Methods to quickly modify an existing table without user interaction:
IWPTextCursor.ExitTable 267
IWPTextCursor.TableSplit 267
IWPTextCursor.TableSort 267
IWPTextCursor.ASetCellProp 268
IWPTextCursor.MergeCellHorz 270
IWPTextCursor.MergeCellVert 270
IWPTextCursor.ASetCellStyle 269

Also available:

TextCommand 208(13) combines all adjacent tables. Returns the count of deleted tables.

TextCommand 208(14) tries to create merged cells for cells which are wider than the others in a column. Returns the count of changed tables.

8.3.20 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 347	IWPAttrInterface.GetWPCSS 285

IWPParInterface.ParWPCSS 347 | IWPAttrInterface.SetWPCSS 290

8.3.21 Spell Check

TextDynamic optionally supports optional integrated spell check.

A) Integrated spell check

This option adds a spell checking feature to the component. It is possible to locate spelling errors using a popup dialog and also to highlight misspelled words with red underlines. Also included is a dictionary compiler to create new dictionaries from word list files.

The method Memo.TextCommand(7, ParamA, 0) 208 can be used to control the spell check.

The following values are allowed for ParamA:
0 =Start SpellCheck (using the optional internal spellchecker),
1 = Start Thesuarus (reserved), 2=Start SpellAsYouGo (internal or external),
3 = Stop SpellAsYouGo, (internal or external),
4 = Show SpellCheckSetup (internal only).

The following "wpa" Actions manage spell check:

This action names can be used with wpaProcess() 122
"DiaSpellOptions" - show the options dialog
"Spellcheck" - start the spell check
"SpellAsYouGo" - can be called with parameter "1" to switch the online spell check (curly underlines) ON or with parameter "0" to switch it OFF. If no parameter is used, the state will be toggled.

Interface SpellCtrl

You can use the interface IWPSpell 388 to load dictionaries from certain directories, select the language which is currently used and load or save the current setup to registry or INI file. In the IWPSpell topic you will find some sample code.

B) Custom spell check

"Custom spell check 108" allows it to use an external data base with words to verify the spelling. We recommend to use the integrated spell check methods instead, because they guarantee high performance and compact dictionaries, but if you need to use an existing spell check solution, TextDynamic allows this, too.

Note: If "custom" spell check is enabled, the internal spell check feature is disabled!

8.4 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.

It contains the load 187 and save 194 methods, methods to enumerate the paragraphs 180 and styles 181, to work with header and footers 182 or images 185. It also contains the method SetBProp 197 which is used to change various options of the editor for viewing and formatting.

It also publishes other interfaces, such as TextCursor 219. This interface contains the method to move the cursor and to create text 247, text boxes 251 and tables 224.

You can use Memo.Statistic 207 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 TextDynamic Version 1.35 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

ActiveText 162
AutoZoom 169
ColorDesktop 169
ColorHighlight 169
ColorHighlightText 170
ColorPaper 170
CurrAttr 162
CurrBand 163
CurrentZooming 163
CurrGroup 163
CurrObj 163
CurrPar 164
CurrParAttr 164
CurrSelAttr 165
CurrSelObj 165
CurrStyle 165
CurrStyleAttr 165
DefaultIOFormat 170
EditingMode 170
Focussed 171
FormCompletion 171
GUIMode 171
Hidden 172
InitialDir 172
LabelDef 166
LastFileName 167
LayoutMode 172
Modified 173
PageSize 167
PageSizeList 167

Methods

Activate 213
AppendOtherText 175
BlockAdd 176
BlockAppend 177
BlockFind 178
Clear 178
ClientToScreen 213
CopyToClipboard 178
CutToClipboard 178
DebugShowParProps 178
DeleteLeadingSpace 179
DeletePage 179
DeleteParWithCondition 179
DeleteStyle 180
DeleteTrailingSpaces 180
EnumDataBlocks 180
EnumParagraphs 180
EnumParSiblings 181
EnumParStyles 181
EnumSelParagraphs 181
EnumTextObj 182
FindFooter 182
FindHeader 182
GetMouseXY 213
GetNumberStyle 183
GetObjAtXY 183
GetPageAsMetafile 184
GetPosAtXY 185
GetRTFVariable 186
GetXY 186

Methods

8.4.1 Properties

8.4.1.1 ActiveText

Declaration
IWPDataBlock 296 ActiveText;

Description
This is a IWPDataBlock 296 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.

8.4.1.2 CurrAttr

Declaration
IWPAttrInterface 276 CurrAttr;

Description
This is the interface to the current writing attribute of the active editor.

Also see TextAttr 162!

8.4.1.3 TextAttr

Declaration
IWPAttrInterface 276 TextAttr;

Description
This is the interface to the change the current writing attribute of the active editor unless text is selected.

In case text has been is selected the properties of the selected text will be updated.

This property is used best to apply and read the text attributes to implement a custom toolbar.

8.4.1.4 DefaultAttr

Declaration

IWPAttrInterface 276 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 353();
Memo.DefaultAttr.SetFontface("Courier New");
```

8.4.1.5 CurrBand

Declaration
WPReportBand 376 CurrBand;

Description
Reserved

8.4.1.6 CurrentZooming

Declaration
```
int CurrentZooming;
```

Description
This is the current zooming value. It is automatically adjusted when auto zooming is enabled.

8.4.1.7 CurrGroup

Declaration
IWPReportBand 376 CurrGroup;

Description
Reserved

8.4.1.8 CurrObj

Declaration
IWPTextObj 396 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 237

This C# code inserts a horizontal line and then changes it's color to red.

```
wpdllInt1.TextCursor.InputObject 242(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 219
Dim Field As IWPTextObj 396
Set Cursor = WPDLLInt1.Memo.TextCursor
Set Field = WPDLLInt1.Memo.CurrObj
```

```
Cursor.InputText "Some text"
Cursor.InputField 237 "FieldName", "Display Text", False
Field.Mode = 2 ' Make it Editable!
Cursor.InputText "Some more text"

'Switch to edit mode
WPDLLInt1.Memo.FormCompletion 171 = True
```

8.4.1.9 CurrPar

Declaration

IWPParInterface 339 CurrPar;

Description

This is the interface to manipulate the current paragraph. This is the paragraph the cursor (insertion marker) is located within.

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 164. If you only need to change the attribute of certain characters use either CharAttr 352 or SetCharAttr 365.

Example:


```
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);
```

8.4.1.10 CurrParAttr

Declaration

IWPAttrInterface 276 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 use either CharAttr 352 or SetCharAttr 365.
This interfaces is also provided as IWPMemo.CurrParAttr.

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);
```

8.4.1.11 CurrSelAttr

Declaration
IWPAttrInterface[276] CurrSelAttr;

Description
This interface can be used to change the attributes of the text which is currently selected.

Also see TextAttr[162]!

8.4.1.12 CurrSelObj

Declaration
IWPTextObj[396] 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");
}
```

8.4.1.13 CurrStyle

Declaration
IWPParInterface[339] 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()[196]

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(191 **true**191, true)191 after you have changed a style.

8.4.1.14 CurrStyleAttr

Declaration

IWPAttrInterface 276 CurrStyleAttr;

Description
This interface is used to change the character attributes defined by the current style. Also see CurrStyle 165 and SelectStyle 196.

VB.NET example:

```
Memo.CurrStyleAttr.SetFontface("Courier New")
Memo.CurrStyleAttr.SetFontSize("12")
```

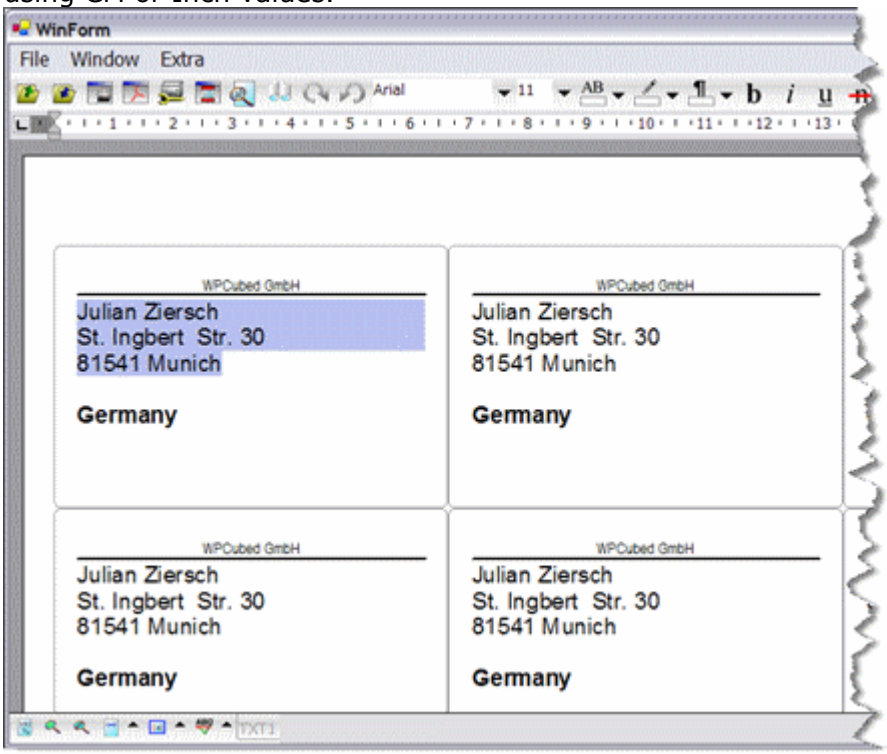
Notes:
- You need to execute ReformatAll(191 **true** 191 , true) 191 after you have changed a style.
- The character attributes defined by a style will be only used by text which does not override these attributes. When you insert new text you can clear all writing attributes using Memo.CurrAttr.Clear() 280.

8.4.1.15 LabelDef

Interface to display labe sheet and change label setup

Declaration
IWPLabelDef 314 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 154

8.4.1.16 LastFileName

Declaration
```
string LastFileName;
```

Description
The last file name which was used for file load or save.

8.4.1.17 PageSize

Declaration
IWPPageSize 333 PageSize;

Description
You can use this interface to modify the page size information stored with the document.

8.4.1.18 PageSizeList

Declaration
IWPPageSizeList 338 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 184 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 191 to update the screen and layout information. (Not required by toggling "Active" or changing the Resolution.

8.4.1.19 PrintParameter

Declaration
IWPPrintParameter 370 PrintParameter;

Description
This interface provides access to special printing options. You can print only the odd or even pages.

8.4.1.20 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 195 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 146 to implement text macros.

Note: To insert RTF or HTML formatted strings instead of SelText use LoadFromString 188.

8.4.1.21 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 195 to retrieve the text as RTF, ANSI, HTML or WPT string.

When Writing:
Load the text from a unicode string. Use LoadFromString 188 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 103 and Text2 104 which always save formatting information and is able to load formatted text.

8.4.1.22 TextCursor

Methods and properties to insert text and change current position.

Declaration
```
IWPTextCursor 219 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 219 for a list of methiods.`

.NET: We recommend to assign the value of "TextCursor" to a variable and call wpdllint.ReleaseInt 123(variable) when the reference is not any longer required.

8.4.1.23 SpecialTextAttr

Declaration
```
IWPCharacterAttr 292 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 292 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 star 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;
```

8.4.1.24 Exclusive in IWPMemo

The following properties are not included in RTF2PDFs IWPEditor.

8.4.1.24 A) AutoZoom

Declaration
```
AutoZoom AutoZoom
```

Description
The current zoom mode. Possible values are:
0: AutoZoomOff
1: AutoZoomWidth
2: AutoZoomFullPage
3: AutoZoomAdjustColumnCount
4: AutoZoomAsManyAsPossibleInRow

The modes 3 and 4 are usually combined with a small Zooming 175 value, i.e. 15%.

Tip: If you need let the text wrap to the bounds of the editor window please switch on the WordWrap 175 mode.

8.4.1.24 B) ColorDesktop

Declaration
```
int ColorDesktop;
```

Description
This is the background color as RGB value.

8.4.1.24 C) ColorHighlight

Applies to

IWPMemo 160

Declaration
```
int ColorHighlight;
```

Description
This is the highlighting color as RGB value.

8.4.1.24 D) ColorHighlightText

Applies to
IWPMemo 160

Declaration
```
int ColorHighlightText;
```

Description
This is the text highlighting color as RGB value.

8.4.1.24 E) ColorPaper

Applies to
IWPMemo 160

Declaration
```
int ColorPaper;
```

Description
This is the "paper" color as RGB value. The paper color will not be printed but you can specify a color in the EMF export routine SavePageAsMetafile 193.

8.4.1.24 F) DefaultIOFormat

Applies to
IWPMemo 160

Declaration
```
int DefaultIOFormat;
```

Description
The DefaultIOFormat controls the way the Save and Load actions operate. You can select a format name and also options, see "FormatStrings 150".

8.4.1.24 G) EditingMode

Applies to
IWPMemo 160

Declaration
```
int EditingMode
```

Description
This property selects several properties at once (quick configuration).

wpconfAsEditor=0: Enabled all editing features and undo. Select full width auto zoom.

wpconfAsPreview=1: Disable all editing features. Select full width auto zoom. Hide spellcheck markers.

wpconfAsThumbnails=2: Disable all editing features - switch to thumbnail zooming mode.

8.4.1.24 H) Focussed

Applies to
IWPMemo 160

Declaration
```
bool Focussed
```

Description
The property is true if the editor has the focus.

8.4.1.24 I) FormCompletion

Applies to
IWPMemo 160

Declaration
```
bool FormCompletion;
```

Description
Enable / Disable the form completion mode. In Form completion mode only the editable (Memo.CurrObj.Mode 163=2) mail merge fields can be edited. The cursor will automatically jump from field to field. The text outside of the fields is protected.

8.4.1.24 J) GUIMode

Applies to
IWPMemo 160

Declaration
```
EditorGUI GUIMode
```

Description
This property selects the elements in the GUI of this editor. It uses this bitfield.

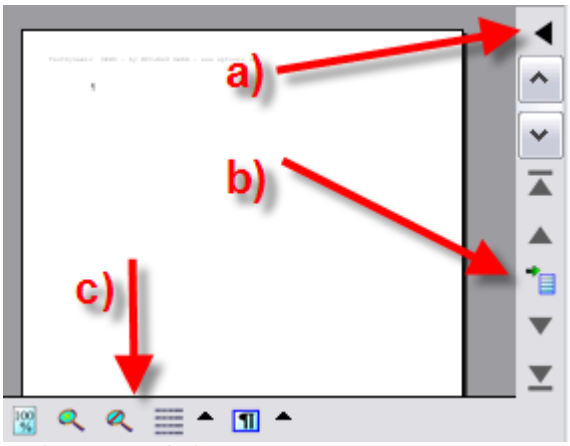
Activate the ruler
wpguiRuler = 2

Activate the vertical scrollbar
wpguiVertScrollBar=4

Activate the horizontal scrollbar
wpguiHorzScrollBar=8

Switch the vertical ruler on and off
wpguiVertRuler =16

Activate the "Gutter" on the left of the editor
wpguiGutter =32



Activate Panel a)
wpguiPanelV1 =64

Activate Panel b)
wpguiPanelV2 =128

Activate Panel c)
wpguiPanelH1 =256

This flag activates the tabset besides of the panel c).
The tabset is used by the SimMDI 46 demo.
wpguiPanelH2 =512

8.4.1.24 K) Hidden

Applies to
IWPMemo 160

Declaration
```
bool Hidden
```

Description
You can assign TRUE to hide the editor. This can be useful if you are creating a report and do not want to display the report template.
"Hidden" only works if TextDynamic uses the two editor mode. Please make sure you set the Hidden property of the other Memo to false.

8.4.1.24 L) InitialDir

Applies to
IWPMemo 160

Declaration
```
string InitialDir;
```

Description
The initail directory used as default for load and save dialogs.

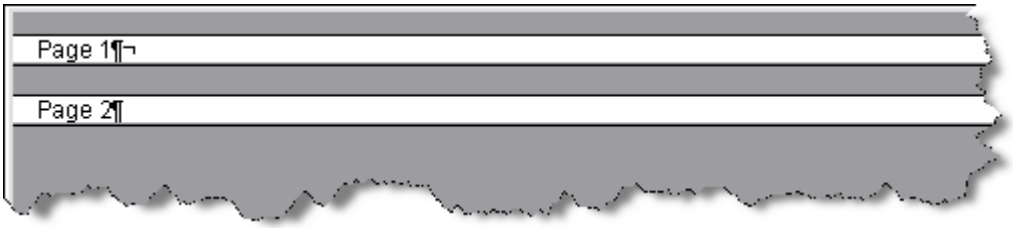
8.4.1.24 M) LayoutMode

Applies to
IWPMemo 160

Declaration
```
LayoutMode LayoutMode
```

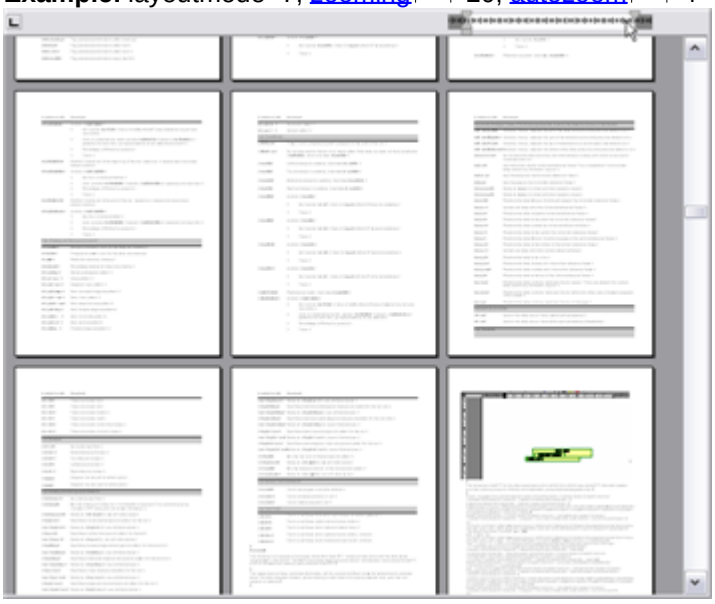
Description
Set the layout mode:

0=wplayNormal - display the text similar to the windows application notepad
1=wpWordWrapView - like 'normal' but also activate word wrap at the right border
2=wplayShowManualPageBreaks - similar to 'normal' but show page breaks as lines



3=wplayPageGap - similar to 'normal' but show page breaks as gaps
4=wplayExtendedPageGap - similar to wplayPageGap but show left and right margins
5=wplayShrunkenLayout - display page withouth the top and bottom margin
6=wplayLayout - display page withouth header and footer texts (but with margins)
7=wplayFullLayout - display full page (WYSIWYG)
8=wplayDualPageView - show 2 pages side by side (full page or full width zoom mode)
9=wplayThumbNailView - display small pages

Example: layoutmode=7, zooming 175=20, autozoom 169=4



Note: You can use the provided **test application** 422 to test the different modes!

8.4.1.24 N) Modified

Applies to
IWPMemo 160

Declaration
```
bool Modified;
```

Description
This property is true if the text was modified. You can also change it in code.

8.4.1.24 O) Readonly

Applies to
IWPMemo 160

Declaration
```
bool Readonly;
```

Description
If this property is true the text is readonly, the user cannot insert or delete text.

If you this property to true also the modifications to table width and object size is disabled.

Tips:

a) You can use SetBProp 197 to adjust the protection i.e. SetBProp(0,10,1) makes the text readonly but the user can still change table widths.

b) To disable the possibility to select text and move the cursor use

```
Memo.SetBProp 2, 0, 1 ' wpDisableSelection
Memo.SetBProp 2, 1, 1 ' wpDisableCaret
```

8.4.1.24 P) ShowFields

Enable/Disable compressed field name display

Declaration
```
bool ShowFields;
```

Description
If this property is true the start marker of merge fields will display the field name, the end marker and the embedded text will be hidden.

This property can also be toggled by the WPA action ShowFieldsEx 419

Example: INTERPRET

Note: The field markers << and >> are usually visible. To hide the markers use
`Memo.SpecialTextAttr(wpInsertpoints).Hidden = True`

Note: In the current version text text can only be completely hidden, if the field text is not longer than one paragraph! This text can be hidden with TextCommand 213 (15)

8.4.1.24 Q) TextReadFormat

Applies to
IWPMemo 160

Declaration
```
string TextReadFormat;
```

Description
Set the default format string 150 used for the Load functions.

Please note that RTF, HTML and WPT format is auto detected.

8.4.1.24 R) TextWriteFormat

Applies to
IWPMemo 160

Declaration
```
string TextWriteFormat;
```

Description
Set the default format string 150 used for the Save functions.

8.4.1.24 S) TopOffset

Applies to
IWPMemo 160

Declaration
```
int TopOffset;
```

Description
This property is modified when the user scrolls the text vertically. It can be also used to scroll, for example to synchronize two editors.

8.4.1.24 T) WordWrap

Declaration
```
bool WordWrap;
```

Description
If this property is true the text will be wrapped according to the bounding box of the editor.

The page width will not be used. Please note that word wrap is always active, this property only changes the X position where the word wrap takes place.

Please also check out the properties:AutoZoom 169 and LayoutMode 172.

8.4.1.24 U) Zooming

Declaration
```
int Zooming
```

Description
This property controls the zooming in percent. Because zooming in TextDynamic is so fast, it is possible to connect this property to a sliding bar - this even works with large texts. When you have selected AutoZoom=AutoZoomAsManyAsPossibleInRow you will see that with smaller zoom values the pages will be arranged horizontally. The text is still editable in this mode.

Please note the zooming only changes the text height, it does not resize the GUI controls and the offsets. You can use SetIProp 206 with ID 5 and 6 to change the X and Y offset (or set it to 0).

8.4.2 Methods

8.4.2.1 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 longtext with multiple letters.
To use AppendOtherText please initialize TextDynamic in the double editor mode, using SetEditorMode(1,...) 114.

Alternatively you can use RTFDataAppendTo 191.

Parameters	
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!)

Category
Mailmerge 154

8.4.2.2 BlockAdd

Creates or selects a header or footer text layer

Declaration
```
IWPDataBlock 296 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 296 reference. This method works similar as TextCursor.InputHeader 240 and TextCursor.InputFooter 239 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() 123 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 123 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 367("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 351(", bold text", -2);
```

8.4.2.3 BlockAppend

Creates a new header or footer layer

Declaration

```
IWPDataBlock 296 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 296 reference so you can modify the properties.

Note: Please don't forget to call ReleaseInt() 123 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)
```

8.4.2.4 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.

Note: Please don't forget to call ReleaseInt() with the returned interface at the end of your code.

8.4.2.5 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.

8.4.2.6 CopyToClipboard

Copy text to clipboard

Declaration
```
void CopyToClipboard();
```

Category
Standard Editing Commands

8.4.2.7 CutToClipboard

Copy text to clipboard and delete it

Declaration
```
void CutToClipboard();
```

Category
Standard Editing Commands

8.4.2.8 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

8.4.2.9 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 180 and DeleteParWithCondition 179.

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.

8.4.2.10 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.

8.4.2.11 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 154

8.4.2.12 DeleteStyle

Declaration
```
void DeleteStyle([In] string StyleName);
```

Description
This method deletes the style with the given name.

Category
Paragraphstyle Support 157

8.4.2.13 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 179 and DeleteParWithCondition 179.

Parameters	
EmptyFields	If true, paragraphs which are empty except for empty merge fields will be also deleted.

8.4.2.14 EnumDataBlocks

Declaration
```
void EnumDataBlocks([In] int EventParam);
```

Description
With EnumDataBlocks the event OnEnumDataBlocks 126 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 148

8.4.2.15 EnumParagraphs

Declaration
```
void EnumParagraphs([In] bool OnlyActiveText, [In] int EventParam);
```

Description
The event OnEnumParOrStyle 127 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.

Category
Callback Functions 148

8.4.2.16 EnumParSiblings

Declaration
```
void EnumParSiblings([In] bool FromStart, [In] int EventParam);
```

Description
Enumerate all paragraphs in current paragraph nesting level and call event OnEnumParOrStyle 127. 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 148

8.4.2.17 EnumParStyles

Declaration
```
void EnumParStyles([In] int EventParam);
```

Description
The event OnEnumParOrStyle 127 will be called for all paragraph styles which are defined.

Parameters	
EventParam	This value will be passed through to the event handler.

Category
Callback Functions 148
Paragraphstyle Support 157

8.4.2.18 EnumSelParagraphs

Declaration
```
int EnumSelParagraphs([In] bool OnlyCompletePars, [In] bool ControlParsToo, [In] int EventParam);
```

Description
The event OnEnumParOrStyle 127 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 148

8.4.2.19 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 127 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 148

8.4.2.20 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 148

8.4.2.21 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 148

8.4.2.22 GetNumberStyle

Declaration
```
IWPNumberStyle 328 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 328 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);
```

Note: Please don't forget to call ReleaseInt() 123 with the returned interface at the end of your code.

8.4.2.23 GetObjAtXY

This method searches for an object at a certain x,y position.

Declaration
```
IWPTextObj 396 IWPTextObj GetObjAtXY([In] int X, [In] int Y, [In] int TypeMask, [In] int Mode);
```

Description
The Result is a reference to the IWPTextObj 396 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.MouseEventArgs e)
{
   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 149

8.4.2.24 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) 191 if the editor was dynamically created.
To create a file use SavePageAsMetafile 193.

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

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.Imaging.Metafile(metaHandle,
                aMetafile.Save("c:\\page__" + i.ToString() + ".emf");
        }
}
temp_editor.DisposeEditor();
```

Category
Load and Save 150

8.4.2.25 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: -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)
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;
}
```

8.4.2.26 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 147

8.4.2.27 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 147

8.4.2.28 LoadFromFile

Declaration

```
bool LoadFromFile([In] string filename, [In] bool Insert, [In] string FormatStr 150);
```

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.

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 209(6,1, dll_path)
LoadFromFile can be also used with a http path to load data from a web server.

Category
Load and Save 150

8.4.2.29 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.

Category
Load and Save

8.4.2.30 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.

Tip: In many cases using LoadFromVar will be more efficient:
```
editorToPreview.Memo.LoadFromVar(
   editorToBePreviewed.Memo.SaveToVar(false, "RTF"),
   false, "RTF");
```

Category
Load and Save

8.4.2.31 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 196.

Note: variant array of bytes work much more efficient to store RTF data than strings.

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. 150

Category
Load and Save 150

8.4.2.32 MergeText

Start the data merging process fo all or a group of fields.

Applies to
IWPMemo 160

Declaration
```
void MergeText([In] string FieldName);
```

Description
Start the data merging process. The event OnFieldGetText 129 will be triggered for each mail merge field.

Mail merge fields can be inserted using IWPTextCursor.InputField(string Name,string Text,bool CursorWithin) 237.

When you need to create a large document from a letter processed by MergeText for all rows in a RecordSet, i.e. to do mass mailing, use the method Memo2.AppendOtherText 175. Please initialize TextDynamic in the double editor mode, using SetEditorMode(1,...) 114.

If you need to create tables with the contents of a complete RecordSet you can consider to use the internal reporting engine 372.

The field markers << and >> are usually visible. To hide them use

```
WPDLLInt1.SpecialTextAttr(wpInsertpoints).Hidden = True
```

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) 233 to convert the text into merge fields.

Parameters	
Name	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 154 and the "mail merge API introduction 72".

8.4.2.33 PasteFromClipboard

Declaration
```
void PasteFromClipboard();
```

Description
Paste text from clipboard.

Category
Standard Editing Commands 156

8.4.2.34 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 190 1,999999.

Note: You can also use Memo.TextCommandStr(11, range). 212 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 156

8.4.2.35 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 156

8.4.2.36 PtrCommand

Declaration
```
int PtrCommand([In] int PtrComID, [In] int PtrValue, [In] int param, [In] string
StrParam);
```

Description
Reserved for custom enhancements.

8.4.2.37 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.

8.4.2.38 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.

8.4.2.39 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 153

8.4.2.40 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 193.

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 153
Mailmerge 154

8.4.2.41 RTFDataDelete

Declaration

```
void RTFDataDelete([In, MarshalAs(UnmanagedType.BStr)] string Name);
```

Description

Deletes a document with a given name.

Please make sure you select 193 another, valid document before the deletion.

We recommend to not delete the last RTFData object but to clear it.

Category

Logical MDI Support 153
Logical MDI Support 153

8.4.2.42 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 191.

Category
Logical MDI Support 153

8.4.2.43 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 184.

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.GetTempFileName();
 string emf_tempfile = tempfile + ".EMF";
 try
 {
     // Format the text, othewise SavePageAsMetafile does not work
     Memo.ReformatAll(true,false);
     // save this page as image
     Memo.SavePageAsMetafile(0,emf_tempfile,8,0);
     // and insert it here
     TextCursor.InputText("Preview of this page :");
     TextCursor.InputImage(emf_tempfile,0);
     Memo.CurrObj.Frame = 1;
     // Scale to 30 % of the original size
     Memo.CurrObj 396.ScaleSize 407(0,0,30);

     // The preview does not contain the image which jsut has been inserted
```

```
        // so recreate the image file and load it in the image aian
        Memo.ReformatAll(false,false);
        Memo.SavePageAsMetafile(0,emf_tempfile,8,0);
        Memo.CurrObj.Contents_LoadFromFile(emf_tempfile);
    }
    finally
    {
        // Delete the temporary files
        System.IO.File.Delete(emf_tempfile);
        System.IO.File.Delete(tempfile);
    }
```

Category
Load and Save

8.4.2.44 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
```

Category
Load and Save

8.4.2.45 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

8.4.2.46 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

8.4.2.47 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 195.

C# example:
```
object data;
data = wpdllInt1.Memo.SaveToVar(false, "");
```

Parameters	
SelectionOnly	true to only save the selected text.
FormatStr	Select the writer and set options. See FormatStrings. 150

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 150

8.4.2.48 SelectStyle

Declaration
```
void SelectStyle(string StyleName);
```

Description
Selects a certain style. The style can be then manipulated through interface CurrStyle 165 and CurrStyleAttr 276.

VB.NET example:
```
'initialize the reference to "Memo 160"
Dim Memo As IWPMemo
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 246 - there we added an VB.NET example which creates a table using merged cells and paragraph styles.

Category
Paragraphstyle Support 157

8.4.2.49 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 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 198
Group 1 : wpEditOptions 198
Group 2 : wpEditOptionsEx 199
Group 3 : wpProtectProp 200
Group 4 : wpClickableCodes 200
Group 5 : wpWriteObjectMode 201
Group 6 : wpViewOptions 201
Group 7 : wpAsWebPage 202
Group 8 : wpFormatOptions 202
Group 9 : wpFormatOptionsEx 203
Group 10 : wpAcceptFilesOptions 204
Group 11 : wpPrintOptions 204
Group 12 : wpClipboardOptions 204
Group 13 : Security Options. 205
Group 14 : Global Security Options 206

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.

8.4.2.49 A) 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 when scrolling
6: Display text as web page on/off
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!)
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.

8.4.2.49 B) 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

8.4.2.49 C) Group 2 : wpEditOptionsEx

Group 2 : Options for Editing - also see "EditOptions 198"

usage: Memo.SetBProp 197(2, X, -1) to deactivate, Memo.SetBProp 197(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).

8.4.2.49 D) 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 edit fields
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 is
15: ppDontProtectAttributes // V5.15 - if defined it is possible to change the attributes of protected text
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 last char is protected
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 objects and HTMLCODE fields.
20: ppBookmarkKeepStructure - Used with ppIsBookmark. When text is deleted which contains bookmarks the bookmark objects are recreated to preserve the logical structure of the document.
21: ppInsertpointKeepStructure - Used with ppIsInsertpoint - reserve the merge fields.

8.4.2.49 E) 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>...

8.4.2.49 F) 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

8.4.2.49 G) Group 6 : wpViewOptions

Group 6 : Change the way the text is displayed in the editor.

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
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 // don't display 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

8.4.2.49 H) Group 7 : wpAsWebPage

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. Currently it is experimental!

usage: Memo.SetBProp(4, X, -1) to deactivate, Memo.SetBProp(4, X, 1) to activate.

0 : activate/deactivate the "as webpage" display.
When "AsWebpage" is activated a different formatting procedure is used to measure the tables like a webbrowser.
Note: You can use TextCommandStr(6,"..") to enable the provided HTTP DLL to load linked images.
Not all text features are supported!
1 : wpNoMinimumWidth
2 : wpNoMinimumHeight
3 : wpLimitToPageWidth

8.4.2.49 I) 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 // - we suggest to enable this flag. Otherwise tables importaed form RTF can look wrong
1: wpNoMinimumCellPadding // Do not add a one pixel padding to all cells
2: wpfDontBreakTables // 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 // must be combined with EditOptionsEx : AllowCursorInRow
8: wpfIgnoreKeep // 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 (WPTools 4 and <5.14 did it so)
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
25: wpfXMLOutlineMode // Debugging mode: Show paragraph tree similar XML in IE
26: wpWriteRightToLeft // activate RTL writing
27: wpAutoWriteRightToLeft // reserved
Troubleshooting flags
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

8.4.2.49 J) 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 WPTools 4 // Alternatively set global variable WPDoNotAddExternalFontLeading := TRUE to // activate this mode for all editors!
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

8.4.2.49 K) 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) 198

usage: Memo.SetBProp 197(10, X, -1) to deactivate, Memo.SetBProp 197(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

8.4.2.49 L) Group 11 : wpPrintOptions

Group 11 : Modify the way printing is performed

usage: Memo.SetBProp 197(11, X, -1) to deactivate, Memo.SetBProp 197(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
12: wpAllColorsAreBlack - print colors in gray scale

8.4.2.49 M) Group 12 : wpClipboardOptions

Group 12 : Modify the clipboard is used. Here also Drag&Drop is customized

usage: Memo.SetBProp 197(12, X, -1) to deactivate, Memo.SetBProp 197(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

8.4.2.49 N) 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.

8.4.2.49 O) 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)
.
```

8.4.2.50 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:

```
// 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

8.4.2.51 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 147

8.4.2.52 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 157

8.4.2.53 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 158

8.4.2.54 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 158

8.4.2.55 TextCommand

Declaration
```
int TextCommand([In] int ComID, [In] int paramA, [In] int paramB);
```

Description
This method has been reserved for custom enhancements.

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 349 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: 7:
Starts and stops the integrated spell check (requires optional "spell" license).
The following values are allowed for ParamA:

0=Start SpellCheck (using the optional internal spellchecker),
1= Start Thesuarus (reserved),
2=Start SpellAsYouGo (internal or external),
3=Stop SpellAsYouGo, (internal or external),
4 = Show SpellCheckSetup (internal only).

Also see "Custom Spellcheck 108" and IWPSpell 388.

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

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 highlighter.

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.

8.4.2.56 TextCommandStr

Declaration

```
string TextCommandStr([In] int ComID, [In] int param, [In] string StrParam);
```

Description
This method has been reserved for custom enhancements.

8.4.2.56 A) 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.

8.4.2.56 B) 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 code page can be provided a first parameter.

8.4.2.56 C) Command ID 3 -& 4 - clipboard format

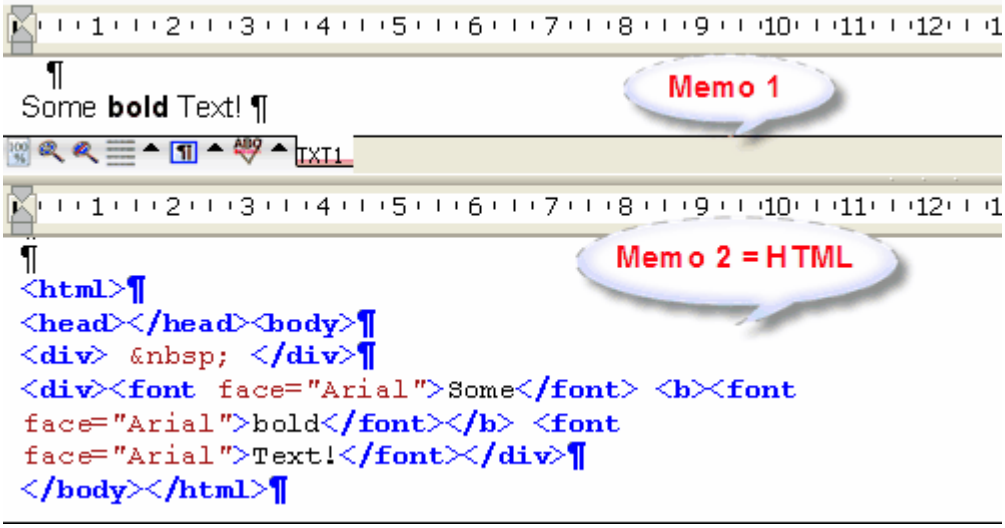
ID 3:
Set the property **TextSaveFormatClipboard**. This property holds the format string which is used to write the text which is copied to the clipboard. It returns the previous value of this property. If param<0 the property is not changed.

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.

8.4.2.56 D) 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[212](9,...)

8.4.2.56 E) Command ID 6 - select HTTP DLL

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 serached in same dirtectory as the main TextDynamic engine. The name is always wphttpget.dll, or with TextDynamic demo: wphttpget_demo.dll. 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 can also activate a new experimental HTML formatting method to format tables differently. Memo.SetBProp(7,0,1) 197.

8.4.2.56 F) 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 413 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 are ignored)
                (CurrPar.ParCommand(1,1,0xFFFF)!=1))
                CurrPar.ParASet((int)WPAT.ParIsOutline,1);
        else CurrPar.ParADel((int)WPAT.ParIsOutline);
        next = Cursor.CPMoveNextPar(false);
}
```

ID 8:
Processes all paragraphs of the body text to select paragraphs to be used as PDF bookmark (outline).

StrParam="" - disables all WPAT.ParIsOutline 413 property.

If the **paragraph starts with** StrParam set the ParIsOutline level provided as Param. If Param = 0 the property will be set to 0.

8.4.2.56 G) Command ID 9 - insert HTML markup

This command inserts HTML open and closing markup.

```
Memo.TextCommandStr(9, 0, "b") creates  <b>|</b>.
```

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  <span
style="font-face:Arial">|</span>.
```

8.4.2.56 H) 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.

8.4.2.56 I) 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 370.

8.4.2.56 J) Command ID 12 - Get Printernames

Return a list of printer names.

8.4.2.56 K) Command ID 13 - BeginPrint

Start the printing cue -
this does the same as the method WPDllInt BeginPrint 106
EndPrint 106

8.4.2.56 L) Command ID 14 - EndPrint

End the printing cue -
this does the same as the method WPDllInt EndPrint 106

8.4.2.56 M) Command ID 15 - Update ShowFields Mode

This commands hides multiple paragraphs within texts in case the property ShowFields 174 = true.

Example: 

8.4.2.57 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;

8.4.2.58 Exclusive in IWPMemo

The following methods are not included in RTF2PDFs IWPEditor.

8.4.2.58 A) Activate

Activates the editor 1 or 2

Applies to
IWPMemo 160

Declaration
```
void Activate();
```

Description
This method will make this editor the current editor. The toolbar will now work with this editor.

8.4.2.58 B) ClientToScreen

Declaration
```
void ClientToScreen([In, Out] ref int X, [In, Out] ref int Y);
```

Description
This function converts editor coordinates into screen coordinates.

Category
Coordinate Conversion 147

RTF2PDF Note: This method is not included IWPEditor

8.4.2.58 C) GetMouseXY

Get mouse page and x,y in twips.

Declaration
```
int GetMouseXY(out int PageNr, out int PageX, out int PageY);
```

Description
This function can be used to calculate the page under the mouse cursor and the X and Y position on the page.

The result value is 0 if there is no page under the mouse cursor, otherwise it can have the following values:

InsidePage=1,
TopMargin=2,
BottomMargin=3,
LeftMargin=4,
RightMargin=5,
AfterTextBody=6,
InsideTextBody=7.

X and Y are measured in twips (inch/1440)

Also see GetXY 186 which retrieves various properties.

Example C#:

```
private void wpdllInt1_MouseMove(object sender, System.Windows.Forms.MouseEventArgs e)
{
 int x,y,p;
 if (wpdllInt1.Memo.GetMouseXY(out p,out x,out y)>0)
 {
  stHint.Text =
     "P" + p.ToString() +
     "X" + x.ToString() +
     "Y" + y.ToString();
 }
}
```

Example VB6:

```
    Dim x As Long
    Dim y As Long
    Dim page As Long
    Memo.GetMouseXY page, x, y
```

Category
Coordinate Conversion 147

RTF2PDF Note: This method is not included IWPEditor

8.4.2.58 D) InsertGraphicDialog

Declaration

```
bool InsertGraphicDialog([In] string InitPath, [In] string Filter, [In] bool AsLink, [In] int PositionMode);
```

Description
Display a file open dialog to insert a graphic file.

Parameters	
InitPath	Directory the dialog starts browsing
Filter	The file open filter, the default is "Graphic Files (*.BMP,*.WMF,*.JPG,*.GIF,*.PNG)|*.BMP;*.WMF;*.EMF;*.JPG;*.JPEG;*.GIF;*.PNG".
AsLink	If true the image will not be embedded, only the file name will be stored with the document.
PositionMode	0=as character, 1=link to paragraph, 2=link to page

Returns
true if an image was inserted. You can use 'CurrObj' to manipulate it.

Category
Image Support 149
Load and Save 150

RTF2PDF Note: This method is not included IWPEditor

8.4.2.58 E) Load

Declaration
```
bool Load([In] string InitPath, [In] string Filter);
```

Description
Display a file open dialog. Optionally provide start directory for dialog and the file filter.

Returns
True if a file was loaded.

Category
Load and Save 150

RTF2PDF Note: This method is not included IWPEditor

8.4.2.58 F) LoadEx

Applies to
IWPMemo 160

Declaration
```
bool LoadEx([In] string InitPath, [In] string Filter);
```

Description
Display a file open dialog. Optionally provide start directory for dialog and the file filter. This method will also use the installed document conversion DLLs. These DLLs are installed by Office, a few are available by standard on Windows XP.

Returns
True if a file was loaded.

Category
Load and Save 150

8.4.2.58 G) Save

Saves a file to known destination.

Declaration
```
bool Save([In] string InitPath, [In] string Filter);
```

Description
You may select a start directory and the file save filter for the save dialog. The dialog will not be displayed if the filename is already known.

Category
Load and Save 150

8.4.2.58 H) SaveAs

Saves a file using a dialog.

Declaration
```
bool SaveAs([In] string InitPath, [In] string Filter);
```

Description

You may select a start directory and the file save filter for the save dialog.

Category
Load and Save 150

8.4.2.58 I) ScreenToClient

Applies to
IWPMemo 160

Declaration
```
void ScreenToClient([In, Out] ref int X, [In, Out] ref int Y);
```

Description
This function converts screen coordinates into editor coordinates.

Category
Coordinate Conversion 147

8.4.2.58 J) ShowDialog

Declaration
```
bool ShowDialog([In] DialogID DialogID, [In] string Param1, [In] string Param2);
```

Description
Display one of the internally defined dialogs. You can select the dialog id and pass 2 string parameters.

RTF2PDF Note: This method is not included IWPEditor

Available IDs:

```
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; // reserved
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;// reserved
DIALOG_PrintBooklet=34;// reserved
DIALOG_DocInfo=35; // // reserved
DIALOG_Templates=36; // // reserved
DIALOG_DocVariable=37; // // reserved
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;// reserved
DIALOG_ManageFormulas=42;// reserved
DIALOG_ManageHeaderFooter=43;
```

8.4.2.58 K) wpaCheck

Declaration

```
int wpaCheck([In] string WPA);
```

Description

Get the checked/enabled states for a certain action. It is usually more effective to use method wpaGetID 218 to calculate the action id once and then work with the flag array provided by wpaGetFlags 120 using the strored index.

Category: Action Names 417

8.4.2.58 L) wpaExec

Declaration

```
bool wpaExec([In] int wpaID, [In] string param);
```

Description

Execute a wpa action using the action ID calculated by wpaGetID 218. The string parameter is used by only a few actions.

Category: Action Names 417

RTF2PDF Note: This method is not included IWPEditor

8.4.2.58 M) wpaGetCaption

Declaration

```
bool wpaGetCaption([In] int wpaID, [In, Out] ref string Name, [In, Out]
ref string Caption, [In, Out] ref string Hint);
```

Description

Retrieve the name, caption and hint for the action with the given id. The result value will be "false" if the id is not defined. To build a list of actions, it is save to try all ids starting with 0 until this function returns false.

Category: Action Names 417

RTF2PDF Note: This method is not included IWPEditor

8.4.2.58 N) wpaGetFlags

Declaration

```
string wpaGetFlags();
```

Description
See method wpaGetFlags 120.

RTF2PDF Note: This method is not included IWPEditor

8.4.2.58 O) wpaGetID

Declaration
```
int wpaGetID([In] string Name);
```

Description
Convert a WPA action name into an action ID (integer). You can find a list of wpa names here: http://www.wpcubed.com/manuals/wpa.htm.

Category: Action Names 417

RTF2PDF Note: This method is not included IWPEditor

8.4.2.58 P) wpaProcess

Declaration
```
bool wpaProcess([In] string wpaName, [In] string param);
```

Description
Excute a wpa action using its name. The string parameter is used by only a few actions. . You can find a list of wpa names here: http://www.wpcubed.com/manuals/wpa.htm.

Category: Action Names 417

RTF2PDF Note: This method is not included IWPEditor

8.4.2.58 Q) wpaSetCaption

Declaration
```
bool wpaSetCaption([In] int wpaID, [In] string Caption, [In] string
Hint);
```

Description
Modify the Caption and Hint used by a certain action. The id is calculated by wpaGetID 218.

Category: Action Names 417

RTF2PDF Note: This method is not included IWPEditor

8.4.2.58 R) wpaSetFlags

Declaration
```
void wpaSetFlags([In] int Start, [In] int Count, [In] string AllFlags);
```

Description

Modify the enable/hidden(checked state of certain actions. This method may be only executed within the event OnUpdateGUI.

Please note that the .NET class implements the method:
```
public void wpaSetFlags 62(int Editor, int Start, int Count, byte[] AllFlags)
```

Since this method works with an array of bytes we used recommend to use it instead of Memo.

wapSetFlags.

Category: Action Names 417

RTF2PDF Note: This method is not included IWPEditor

8.5 IWPTextCursor

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.

Use **Memo.TextCursor** 168 to get a reference to this powerful and important interface.

Please note, the text box and foot note support has to be activated using SetEditorMode 114!

Properties

- CPCellPtr 220
- CPLineNr 220
- CPObjPtr 220
- CPPageLineNr 221
- CPPageNr 221
- CPParNr 221
- CPParPtr 221
- CPPosInLine 221
- CPPosInPar 222
- CPPosition 222
- CPRowPtr 222
- CPStylePtr 222
- CPTableColNr 223
- CPTablePtr 223
- CPTableRowNr 223
- IsSelected 223
- PageCount 223

Methods

- AddTable 224
- AppendRow 225
- CheckState 225
- Clear 226
- CombineCellHorz 226
- CombineCellVert 227
- CPMoveAfterTable 227
- CPMoveBack 227
- CPMoveBeforeTable 227
- CPMoveNext 228
- CPMoveNextBand 228
- CPMoveNextCell 228
- CPMoveNextGroup 228
- CPMoveNextObject 228
- CPMoveNextPar 229
- CPMoveNextRow 229
- CPMoveNextTable 230
- CPMoveParentTable 230
- CPMovePrevCell 230
- CPMovePrevObject 230
- CPMovePrevPar 230
- CPMovePrevRow 231
- CPMovePrevTable 231
- CPMoveToGroup 231
- CPOpenObj 231
- EnumOpenObj 266
- Delete 232
- DisableProtection 232
- DisableUndo 232
- EnabledProtection 232

Methods

- GetParName 233
- GotoBody 234
- GotoEnd 234
- GotoStart 234
- HideSelection 234
- InputBookmark 235
- InputCalculatedField 235
- InputCell 235
- InputCustomHTML 236
- InputEmbeddedData 236
- InputField 237
- InputFieldObject 238
- InputFooter 239
- InputFootnote 239
- InputHeader 240
- ClearAllHeaders 271
- ClearAllFooters 271
- InputHTML 241
- InputHyperlink 241
- InputImage 241
- InputObject 242
- InputPagebreak 243
- InputParagraph 244
- InputPicture 244
- InputPictureStream 244
- InputRowEnd 245
- InputRowStart 246
- InputSection 247

Methods

- MarkerCollect 253
- MarkerCollectAll 254
- MarkerCPPosition 254
- MarkerDrop 254
- MarkerGoto 254
- MarkerSelect 255
- MovePosition 255
- MoveToBookmark 255
- MoveToTable 256
- Redo 257
- ReplaceText 257
- ReportConvertTable 257
- ReportConvertText 258
- ReportInputBand 258
- ReportInputGroup 258
- ScrollToCP 259
- SelectAll 259
- SelectLine 259
- SelectParagraph 259
- SelectTable 260
- SelectTableColumn 260
- SelectTableRow 260
- SelectText 260

8.5.1 Properties

8.5.1.1 CPCellPtr

Applies to
IWPTextCursor 219

Declaration
```
int CPCellPtr;
```

Description
This is a low level paragraph reference to the cell the cursor is located within.

8.5.1.2 CPLineNr

Applies to
IWPTextCursor 219

Declaration
```
int CPLineNr;
```

Description
This is the number of the current line from start of the text.

Category
Display Status Information 157

8.5.1.3 CPObjPtr

Applies to
IWPTextCursor 219

Declaration
```
int CPObjPtr;
```

Description
This is a low level text object reference to the object at cursor position.

8.5.1.4 CPPageLineNr

Applies to
IWPTextCursor 219

Declaration
```
int CPPageLineNr;
```

Description
This is the number of the current line from start of the page.

8.5.1.5 CPPageNr

Applies to
IWPTextCursor 219

Declaration
```
int CPPageNr;
```

Description
This is the number of the current page.

Category
Display Status Information 157

8.5.1.6 CPParNr

Applies to
IWPTextCursor 219

Declaration
```
int CPParNr;
```

Description
This is the number of the current paragraph.

8.5.1.7 CPParPtr

Applies to
IWPTextCursor 219

Declaration
```
int CPParPtr;
```

Description
This is a low level paragraph reference to the paragraph the cursor is located within.

8.5.1.8 CPPosInLine

Applies to
IWPTextCursor 219

Declaration
```
int CPPosInLine;
```

Description

This is the position in the current line.

Category
Display Status Information 157

8.5.1.9 CPPosInPar

Applies to
IWPTextCursor 219

Declaration
```
int CPPosInPar;
```

Description
This is the position in the current paragraph.

8.5.1.10 CPPosition

Applies to
IWPTextCursor 219

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 157

8.5.1.11 CPRowPtr

Applies to
IWPTextCursor 219

Declaration
```
int CPRowPtr;
```

Description
This is a low level paragraph reference to the row the cursor is located within.

8.5.1.12 CPStylePtr

Applies to
IWPTextCursor 219

Declaration
```
int CPStylePtr;
```

Description
This is the number of the style the current paragraph uses. You can assign 0 to clear the style reference.

8.5.1.13 CPTableColNr

Applies to
IWPTextCursor 219

Declaration
```
int CPTableColNr;
```

Description
This is the number of the current column.

Category
Table Support 158

8.5.1.14 CPTablePtr

Applies to
IWPTextCursor 219

Declaration
```
int CPTablePtr;
```

Description
This is a low level paragraph reference to the table the cursor is located within.

Category
Table Support 158

8.5.1.15 CPTableRowNr

Applies to
IWPTextCursor 219

Declaration
```
int CPTableRowNr;
```

Description
This is the nbumber of the current row. Y

Category
Table Support 158

8.5.1.16 IsSelected

Applies to
IWPTextCursor 219

Declaration
```
bool IsSelected;
```

Description
This property is true if text is selected.

8.5.1.17 PageCount

Applies to
IWPTextCursor 219

Declaration
```
int PageCount;
```

Description
The count of pages in the document - readonly.

Also see category "Display Status Information 157"

8.5.2 Methods

8.5.2.1 IWPTextCursor.AddTable

Create a table and use callback event

Applies to
IWPTextCursor 219

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 125 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 228 and CPMoveNextRow 229. **To create text after the table use ExitTable 267 or InputParagraph(2,"") 244 !**

Parameters	
Name	An optional name for the table. You can use MoveToTable 256 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 268 and ASetCellStyle 269 to quickly modify the properties, text and styles of certain cells in an existing table.

Category

Callback Functions 148
Table Support 158

8.5.2.2 IWPTextCursor.AppendRow

Applies to
IWPTextCursor 219

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 158

8.5.2.3 IWPTextCursor.CheckState

Applies to
IWPTextCursor 219

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.

8.5.2.4 IWPTextCursor.Clear

Applies to
IWPTextCursor 219

Declaration
```
procedure Clear;
```

Description
Clears the current text layer (data block) - does not clear number- or paragraph styles.

8.5.2.5 IWPTextCursor.CombineCellHorz

Applies to
IWPTextCursor 219

Declaration
```
function CombineCellHorz: WordBool;
```

Description
Merges the selected cells horizontally. Also see IWPTextCursor.MergeCellHorz 270.

Category

Table Support 158

8.5.2.6 IWPTextCursor.CombineCellVert

Applies to
IWPTextCursor 219

Declaration
```
function CombineCellVert: WordBool;
```

Description
Merges the selected cells vertically. Also see IWPTextCursor.MergeCellVert 270.

Category
Table Support 158

8.5.2.7 IWPTextCursor.CPMoveAfterTable

Applies to
IWPTextCursor 219

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 158

8.5.2.8 IWPTextCursor.CPMoveBack

Applies to
IWPTextCursor 219

Declaration
```
function CPMoveBack: WordBool;
```

Description
Moves the cursor back one character. If this was successful return true, otherwise false.

8.5.2.9 IWPTextCursor.CPMoveBeforeTable

Applies to
IWPTextCursor 219

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 158

8.5.2.10 IWPTextCursor.CPMoveNext

Applies to
IWPTextCursor 219

Declaration
```
function CPMoveNext: WordBool;
```

Description
Moves the cursor one character to the right. If this was successful return true, otherwise false.

8.5.2.11 IWPTextCursor.CPMoveNextBand

Applies to
IWPTextCursor 219

Declaration
```
function CPMoveNextBand: WordBool;
```

Description
__Reserved__

8.5.2.12 IWPTextCursor.CPMoveNextCell

Applies to
IWPTextCursor 219

Declaration
```
function CPMoveNextCell: WordBool;
```

Description
Move to the cell to the right. If in last cell return false.

Category
Table Support 158

8.5.2.13 IWPTextCursor.CPMoveNextGroup

Applies to
IWPTextCursor 219

Declaration
```
function CPMoveNextGroup: WordBool;
```

Description
__Reserved__

8.5.2.14 IWPTextCursor.CPMoveNextObject

Applies to
IWPTextCursor 219

Declaration
```
function CPMoveNextObject(ObjType: TxTextObjTypes; DontSkipCloseTags: WordBool):
WordBool;
```

Description
Locate next object of a given object type.

8.5.2.15 IWPTextCursor.CPMoveNextPar

Applies to
IWPTextCursor 219

Declaration
```
function CPMoveNextPar(CreateIfNotExists: WordBool): WordBool;
```

Description
Move to next paragraph.

8.5.2.16 IWPTextCursor.CPMoveNextRow

Applies to
IWPTextCursor 219

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();
```

Category
Table Support 158

8.5.2.17 IWPTextCursor.CPMoveNextTable

Applies to
IWPTextCursor 219

Declaration
```
function CPMoveNextTable: WordBool;
```

Description
Move to the next table after the current.

Category
Table Support 158

8.5.2.18 IWPTextCursor.CPMoveParentTable

Applies to
IWPTextCursor 219

Declaration
```
function CPMoveParentTable: WordBool;
```

Description
If in a nested table move to parent table.

Category
Table Support 158

8.5.2.19 IWPTextCursor.CPMovePrevCell

Applies to
IWPTextCursor 219

Declaration
```
function CPMovePrevCell: WordBool;
```

Description
Move to the cell to the left. If in last cell return false.

8.5.2.20 IWPTextCursor.CPMovePrevObject

Applies to
IWPTextCursor 219

Declaration
```
function CPMovePrevObject(ObjType: TxTextObjTypes): WordBool;
```

Description
Locate previous object of a given object type.

8.5.2.21 IWPTextCursor.CPMovePrevPar

Applies to

IWPTextCursor 219

Declaration
```
function CPMovePrevPar: WordBool;
```

Description
Move to previous paragraph.

8.5.2.22 IWPTextCursor.CPMovePrevRow

Applies to
IWPTextCursor 219

Declaration
```
function CPMovePrevRow: WordBool;
```

Description
Move to previous row. If at start of table return false.

8.5.2.23 IWPTextCursor.CPMovePrevTable

Applies to
IWPTextCursor 219

Declaration
```
function CPMovePrevTable: WordBool;
```

Description
Move to previos table before the current table.

8.5.2.24 IWPTextCursor.CPMoveToGroup

Applies to
IWPTextCursor 219

Declaration
```
function CPMoveToGroup(const Name: WideString; InsideOfCurrentGroup: WordBool;
CreateIfNotExist: WordBool; BandElement: Integer): WordBool;
```

Description
__Reserved__

8.5.2.25 IWPTextCursor.CPOpenObj

Applies to
IWPTextCursor 219

Declaration
```
function CPOpenObj(ObjTypMask: Integer): IWPTextObj 396;
```

Description
Checks which objects are open at the current position. Returns a IWPTextObj interface or null if no object is found. Use ObjTypMask 399=0 it will find any open object, otherwise the specified object type must match.

Note: Please don't forget to call ReleaseInt() 123 with the returned interface at the end of your code.

8.5.2.26 IWPTextCursor.Delete

Applies to
IWPTextCursor 219

Declaration
```
procedure Delete(CharCount: Integer);
```

Description
Delete "CharCount" characters at the current position. Will also delete paragraph breaks.

8.5.2.27 IWPTextCursor.DisableProtection

Applies to
IWPTextCursor 219

Declaration
```
procedure DisableProtection;
```

Description
Disable the protection of the text temporarily. Enable the protection with EnableProtection.

8.5.2.28 IWPTextCursor.DisableUndo

Applies to
IWPTextCursor 219

Declaration
```
procedure DisableUndo;
```

Description
Disable undo temporarily.

Category
Standard Editing Commands 156

8.5.2.29 IWPTextCursor.EnabledProtection

Applies to
IWPTextCursor 219

Declaration
```
procedure EnabledProtection;
```

Description
Use after DisableProtection to enable to protect the text again.

Category
Standard Editing Commands 156

8.5.2.30 IWPTextCursor.EnableUndo

Applies to
IWPTextCursor 219

Declaration
```
procedure EnableUndo;
```

Description
Enable undo after it was dsiabled with DisableUndo.

Category
Standard Editing Commands 156

8.5.2.31 IWPTextCursor.FieldsFromTokens

Applies to
IWPTextCursor 219

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>>. This method does the same as "FieldsFromTokens" 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 154

8.5.2.32 IWPTextCursor.FindText

Applies to
IWPTextCursor 219

Declaration
```
procedure FindText(const Text: WideString; FromStart: WordBool; CaseSensitive:
WordBool; MoveCursor: WordBool; WholeWords: WordBool; var Found: WordBool);
```

Description
Locates a text

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 serach csae sensitive
MoveCursor	TRUE to move the cursor to the found position. Otherwise the text will be selected but the cursor will not be moved.
WholeWords	TRUE to find whole words only.
Found	Will be set to TRUE if the text was found.

8.5.2.33 IWPTextCursor.GetParName

Applies to
IWPTextCursor 219

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.

8.5.2.34 IWPTextCursor.GotoBody

Applies to
IWPTextCursor 219

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 148

8.5.2.35 IWPTextCursor.GotoEnd

Applies to
IWPTextCursor 219

Declaration
```
procedure GotoEnd;
```

Description
Move cursor to the end of the text.

8.5.2.36 IWPTextCursor.GotoStart

Applies to
IWPTextCursor 219

Declaration
```
procedure GotoStart;
```

Description
Move cursor to the start of the text.

8.5.2.37 IWPTextCursor.HideSelection

Applies to
IWPTextCursor 219

Declaration

```
procedure HideSelection;
```

Description
Hide (not clear) the selection.

8.5.2.38 IWPTextCursor.InputBookmark

Applies to
IWPTextCursor 219

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 149

8.5.2.39 IWPTextCursor.InputCalculatedField

Applies to
IWPTextCursor 219

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

8.5.2.40 IWPTextCursor.InputCell

Applies to
IWPTextCursor 219

Declaration
```
void InputCell(string Text, sting StyleName);
```

Description
Creates a new Cell in the row which was started with InputRowStart 246. 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 98.

Parameters
Text : Initialization text for this cell. **You can use CurrPar 164 to modify the cell.**
Stylename : Name of the paragraph style this cell should use.

Example:

Create a table with inserted mail merge 72 fields.

```
IWPMemo 160 Memo;
```

```
Memo = WPDLLInt1.Memo;
IWPTextCursor 219 TextCursor = Memo.TextCursor;
IWPAttrInterface 276 CurrParAttr = Memo.CurrParAttr;
IWPParInterface 339 par = Memo.CurrPar;

TextCursor.InputTable 249(0, "data");

For (int r = 1; r <= 10; r++)
{
    TextCursor.InputRowStart 246(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 237("NR_A", "NR_A_TEXT", false);
                                if (c == 1) TextCursor.InputField("NR_B", "NR_A_TEXT", false);
                                if (c == 2) TextCursor.InputField("DESCRIPTION", "DESCRIPTIONTEXT", false);
                                par.ParASet 362((int)WPAT 410.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", false)
                                else par.IsColMerge 343 = true;

                                par.ParASet((int)WPAT.BorderType, 15);
                                par.ParASet((int)WPAT.BorderWidth, 25);
                                par.ParASet((int)WPAT.BorderWidth, 25);
                            }
      }
      TextCursor.InputRowEnd 245();
}
TextCursor.InputParagraph 244(0,"");
Memo.Reformat();
```

Note: In this example we create fields in each cell using InputField. When you merge long texts suing the event OnFieldGetText 129 event you will need Contents.ContinueOptions 311(256) to make sure the paragraph attributes of the inserted text are preserved.

Category
Table Support 158

8.5.2.41 IWPTextCursor.InputCustomHTML

Applies to
IWPTextCursor 219

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.

8.5.2.42 IWPTextCursor.InputEmbeddedData

Applies to
IWPTextCursor 219

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 405 and IWPTextObj.LoadFromFile 405, these methods will update the attached data, not the image. The method Contents_LoadFromFile 402 however will update the image data.

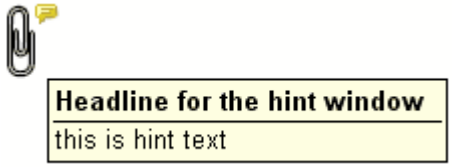
Parameters	
imagefilename	The image file, may be BMP, WMF, EMF, JPG or PNG format. You can also use "pin", "graph", "tag" and "clip" to create an icon. The icon will be only displayed by Acrobat Reader after the export to PDF.
Mode	If bit 1 is set the image will be linked, not embedded. If bit 2 is set 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 163 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 the hint window\"",
   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));
```

8.5.2.43 IWPTextCursor.InputField

Applies to
IWPTextCursor 219

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 163. 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 171" Mode.

Please also see Category Mailmerge 154 and the "mail merge API introduction 72".

8.5.2.44 IWPTextCursor.InputFieldObject

Applies to
IWPTextCursor 219

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 134.

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 419 are available. They can be used with wpaProcess 122 or in the menus defined inside the PCC 424 file.

InsertTextFieldPAGE,
InsertTextFieldNEXTPAGE,
InsertTextFieldPRIORPAGE,
InsertTextFieldNUMPAGES,
InsertTextFieldSECTIONPAGES,
InsertTextFieldDATE,
InsertTextFieldTIME,

Note:
Also see function InputObject() 242.

8.5.2.45 IWPTextCursor.InputFooter

Applies to
IWPTextCursor 219

Declaration
```
void InputFooter(int Range, string name, string Text)
```

Description
Create a footer with the specified "Range" selection. Please also see InputHeader 240 and BlockAdd 176.

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.

8.5.2.46 IWPTextCursor.InputFootnote

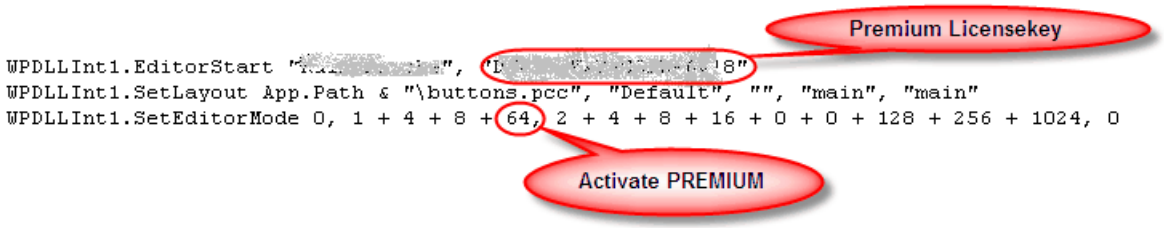
Applies to

IWPTextCursor 219

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 53, 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.

8.5.2.47 IWPTextCursor.InputHeader

Applies to
IWPTextCursor 219

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() 239.

Please also see Memo.BlockAdd 176.

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 238

8.5.2.48 IWPTextCursor.InputHTML

Applies to
IWPTextCursor 219

Declaration
```
procedure InputHTML(const Text: WideString);
```

Description
Inserts HTML formatted text at cursor position.

8.5.2.49 IWPTextCursor.InputHyperlink

Applies to
IWPTextCursor 219

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 163 can be used to change link properties.

Category
Hyperlinks and Bookmarks 149

8.5.2.50 IWPTextCursor.InputImage

Applies to
IWPTextCursor 219

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 163 can be used to change image properties.

Category
Image Support 149

8.5.2.51 IWPTextCursor.InputObject

Applies to
IWPTextCursor 219

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 238, for mail merge fields use InputField 237.

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 or text box and 13 for horizontal lines.
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 134 to provide the text which should be actually displayed.
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 163 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
}
```

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);
```

8.5.2.52 IWPTextCursor.InputPagebreak

Applies to
IWPTextCursor 219

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");
```

8.5.2.53 IWPTextCursor.InputParagraph

Applies to
IWPTextCursor 219

Declaration
```
procedure InputParagraph(Mode: Integer; const StyleName: WideString);
```

Description
Creates a new paragraph and places the cursor on it.

Parameters	
Mode	Bits: 1 : Start a new page before new paragraph 2 : If currently in table append the new paragraph after the table. (Alternative: IWPTextCursor.ExitTable) 267
StyleName	Paragraph style used by new paragraph

Category
Paragraphstyle Support 157

8.5.2.54 IWPTextCursor.InputPicture

Applies to
IWPTextCursor 219

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 244 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

Returns
0 if not successful. Otherwise the interface CurrObj 163 can be used to change imaget properties.

Category
Image Support 149

8.5.2.55 IWPTextCursor.InputPictureStream

Applies to
IWPTextCursor 219

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 163 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);
```

8.5.2.56 IWPTextCursor.InputRowEnd

Applies to
IWPTextCursor 219

Declaration
```
procedure InputRowEnd;
```

Description
Closes a row which was started with InputRowStart 246.

Category
Table Support 158

8.5.2.57 IWPTextCursor.InputRowStart

Applies to
IWPTextCursor 219

Declaration
```
procedure InputRowStart(Mode: Integer);
```

Description
Start a new row. This is only possible after InputTable 249 or after InputRowEnd 245.

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 158

8.5.2.58 IWPTextCursor.InputSection

Applies to
IWPTextCursor 219

Declaration
IWPPageSize 333 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!

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 336/GetProp 336) to set and read it will be automatically set when a property is changed.

Returns
Reference to IWPPageSize 333 interface to modify the page size used by this section.

8.5.2.59 IWPTextCursor.InputString

Applies to
IWPTextCursor 219

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 251) >0 to use the specified CharacterAttr index.

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);
```

8.5.2.60 IWPTextCursor.InputTable

Creates and fills a table not using a callback

Applies to
IWPTextCursor 219

Declaration
```
void InputTable(int Width, string Name);
```

Description
Creates a table object. Create each row with InputRowStart 246 followed by some InputCell 235 and finished by InputRowEnd 245. Use InputParagraph 244 to create text after the table. You can use MoveToTable 256 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 244 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 98 and CurrParAttr 98 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 158

8.5.2.61 IWPTextCursor.InputTabstop

Applies to
IWPTextCursor 219

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 367.

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 IWPMemo
      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)
```

8.5.2.62 IWPTextCursor.InputText

Applies to
IWPTextCursor 219

Declaration
```
procedure InputText(const Text: WideString);
```

Description
Writes some text suing current writing attributes. You can use InputString 248 if you want to specify a character attribute index.

8.5.2.63 IWPTextCursor.InputTextbox

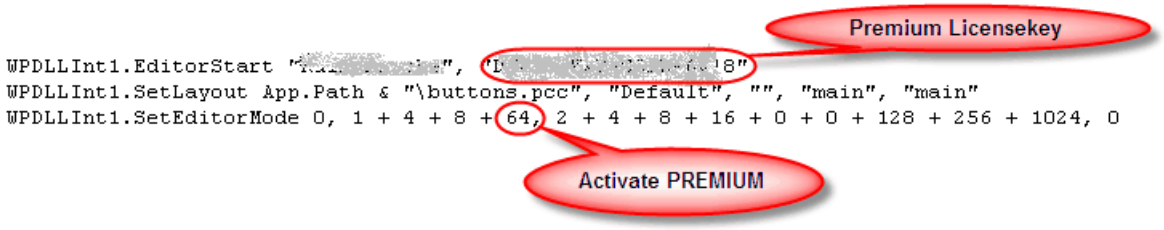
Applies to
IWPTextCursor 219

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 251 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;
Please note that after GotoBody 234 the interface CurrObj cannot be used anymore.

Unless the text box support (with premium license or demo) was activated with SetEditorMode 53, no textbox will be created!



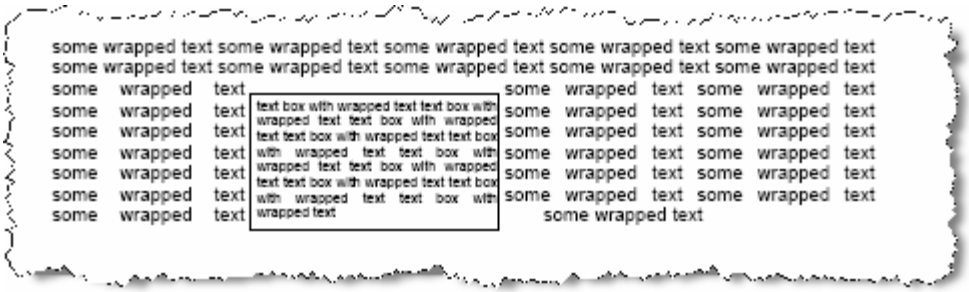
Parameters:
Width : The Width of the box in twips. The height is always determined by the contained text.
Text : The initialization text - may contain HTML or RTF tags. To create an empty box use "CLEAR". If no text was specified (except "CLEAR") the cursor will be places inside the new text box. You can fill in text and then call GotoBody when you are done.

Returns
0 if not successful. Otherwise the interface CurrObj 163 can be used to change object properties.

Tips:
Using CurrObj.property Wrap 400 you can change the way the text wraps around the box.

RelX 400 and RelY 400 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 399 = 2

Example:




```
TextCursor.InputTextbox(2880, "");
// Save writing mode, then enter some text
int save_ch = Memo.CurrAttr.CharAttrIndex;
Memo 160.CurrAttr 276.SetFontSize(8);
for (int i = 0; i < 10; i++)
TextCursor.InputText("text box with wrapped text ");
Memo.CurrPar.Alignment = 3;  // justified text
Memo.CurrObj 396.RelX = 3500;
Memo.CurrObj.RelY = 720;
Memo.CurrObj.Wrap = 4;
TextCursor.GotoBody();
Memo.CurrAttr.CharAttrIndex = save_ch;
Memo.CurrPar 339.Alignment = 3;
for (int i = 0; i < 30; i++)
TextCursor.InputText("some wrapped text ");
```

VB Example:

```
Dim Memo As IWPMemo
Dim Cursor As IWPTextCursor 396
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
```

8.5.2.64 IWPTextCursor.InsertRow

Applies to
IWPTextCursor 219

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 339 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();
```

8.5.2.65 IWPTextCursor.MarkerCollect

Applies to
IWPTextCursor 219

Declaration
```
procedure MarkerCollect(ID: Integer);
```

Description
Remove the marker with the given ID.

Category
Position Markers 154

8.5.2.66 IWPTextCursor.MarkerCollectAll

Applies to
IWPTextCursor 219

Declaration
```
procedure MarkerCollectAll;
```

Description
Remove all markers.

Category
Position Markers 154

8.5.2.67 IWPTextCursor.MarkerCPPosition

Applies to
IWPTextCursor 219

Declaration
```
function MarkerCPPosition(ID: Integer): Integer;
```

Description
Calculate the character position which matches the marker with the given ID.

Category
Position Markers 154

8.5.2.68 IWPTextCursor.MarkerDrop

Applies to
IWPTextCursor 219

Declaration
```
function MarkerDrop(Mode: Integer): Integer;
```

Description
Insert a marker and return the ID.

Category
Position Markers 154

8.5.2.69 IWPTextCursor.MarkerGoto

Applies to
IWPTextCursor 219

Declaration
```
procedure MarkerGoto(Collect: WordBool; ID: Integer);
```

Description
Move the cursor to a marker, optionally collect it.

Category
Position Markers 154

8.5.2.70 IWPTextCursor.MarkerSelect

Applies to
IWPTextCursor 219

Declaration
```
procedure MarkerSelect(StartID: Integer; EndID: Integer);
```

Description
Select the text between marker with ID#1 and ID#2.

Category
Position Markers 154

8.5.2.71 IWPTextCursor.MovePosition

Applies to
IWPTextCursor 219

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.

8.5.2.72 IWPTextCursor.MoveToBookmark

Applies to
IWPTextCursor 219

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 149

8.5.2.73 IWPTextCursor.MoveToField

Applies to
IWPTextCursor 219

Declaration
bool MoveToField(string Name, int Mode, int Option);

This is a versatile methods to jump to and between mail merge fields:

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

If bit 1 was set in parameter "option" the cursor will be positioned after the field object.

This example will move to all fields (from current position) and select the text.

```
if(wpdllInt1.Memo.TextCursor.MoveToField("",1,1))
          wpdllInt1.Memo.CurrObj.Select 407(2)
```

8.5.2.74 IWPTextCursor.MoveToObject

Applies to
IWPTextCursor 219

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.

8.5.2.75 IWPTextCursor.MoveToTable

Applies to
IWPTextCursor 219

Declaration

```
bool MoveToTable(string Name);
```

Description

Moves to a table with a given name. The methods AddTable 224 and InputTable 249 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 268 and ASetCellStyle 269 to quickly modify the properties, text and styles of certain cells in an existing table.

8.5.2.76 IWPTextCursor.Redo

Applies to
IWPTextCursor 219

Declaration
```
function Redo: WordBool;
```

Description
Undoes the last UNDO action.

Category
Standard Editing Commands 156

8.5.2.77 IWPTextCursor.ReplaceText

Applies to
IWPTextCursor 219

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

8.5.2.78 IWPTextCursor.ReportConvertTable

Applies to

IWPTextCursor 219

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 94

8.5.2.79 IWPTextCursor.ReportConvertText

Applies to
IWPTextCursor 219

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 94

8.5.2.80 IWPTextCursor.ReportInputBand

Applies to
IWPTextCursor 219

Declaration
```
procedure ReportInputBand(Typ: Integer; const Name: WideString; Options:
Integer);
```

Description
__RESERVED__

Category
Reporting 94

8.5.2.81 IWPTextCursor.ReportInputGroup

Applies to
IWPTextCursor 219

Declaration
```
procedure ReportInputGroup(const Name: WideString; Options: Integer; Were:
Integer);
```

Description
__RESERVED__

Category
Reporting 94

8.5.2.82 IWPTextCursor.ScrollToCP

Applies to
IWPTextCursor 219

Declaration
```
void ScrollToCP();
```

Description

Scrolls to the current paragraph.

8.5.2.83 IWPTextCursor.SelectAll

Applies to
IWPTextCursor 219

Declaration
```
procedure SelectAll;
```

Description
Select the complete text

Category
Standard Editing Commands 156

8.5.2.84 IWPTextCursor.SelectLine

Applies to
IWPTextCursor 219

Declaration
```
procedure SelectLine;
```

Description
Select the current line

Category
Standard Editing Commands 156

8.5.2.85 IWPTextCursor.SelectParagraph

Applies to
IWPTextCursor 219

Declaration
```
procedure SelectParagraph;
```

Description
Select the current paragraph

Category
Standard Editing Commands 156

8.5.2.86 IWPTextCursor.SelectTable

Applies to
IWPTextCursor 219

Declaration
```
procedure SelectTable;
```

Description
Select the current table

Category
Standard Editing Commands 156
Table Support 158

8.5.2.87 IWPTextCursor.SelectTableColumn

Applies to
IWPTextCursor 219

Declaration
```
procedure SelectTableColumn;
```

Description
Select the current table column

Category
Table Support 158

8.5.2.88 IWPTextCursor.SelectTableRow

Applies to
IWPTextCursor 219

Declaration
```
procedure SelectTableRow;
```

Description
Select the current table row

Category
Table Support 158

8.5.2.89 IWPTextCursor.SelectText

Applies to
IWPTextCursor 219

Declaration
```
procedure SelectText(SelStart: Integer; SelEnd: Integer);
```

Description
Select text range.

Parameters	
SelStart	Select from here

SelEnd	to here (not including)

8.5.2.90 IWPTextCursor.SetColWidth

Applies to
IWPTextCursor 219

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

8.5.2.91 IWPTextCursor.SetParName

Applies to
IWPTextCursor 219

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 256.

3: Assigns the name to the parent-parent table of this paragraph. Returns false if not in a nested table.

Note: CurrPar.ParStrCommand 363 can also change table or paragraph names.

Also see: IWPTextCursor.GetParName 233

8.5.2.92 IWPTextCursor.SetRowHeight

Applies to
IWPTextCursor 219

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).

8.5.2.93 IWPTextCursor.SetTableLeftRight

Change width and margins.

Applies to
IWPTextCursor 219

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 158

8.5.2.94 IWPTextCursor.TableClear

Initialize a table.

Applies to
IWPTextCursor 219

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 158

8.5.2.95 IWPTextCursor.TableDelete

Applies to
IWPTextCursor 219

Declaration
```
function TableDelete: WordBool;
```

Description
Deletes the current table.

Category
Table Support 158

8.5.2.96 IWPTextCursor.Undo

Applies to
IWPTextCursor 219

Declaration
```
function Undo: WordBool;
```

Description
Undeos the last change to the text.

Category
Standard Editing Commands 156

8.5.2.97 IWPTextCursor.UndoClear

Applies to
IWPTextCursor 219

Declaration
```
procedure UndoClear;
```

Description
Clears the undo buffer.

Category
Standard Editing Commands 156

8.5.2.98 IWPTextCursor.WordCharAttr

Applies to
IWPTextCursor 219

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 146

8.5.2.99 IWPTextCursor.WordEnum

Applies to
IWPTextCursor 219

Declaration
```
function WordEnum(const AWord: WideString; CaseSensitive: WordBool;
OnlyActiveText: WordBool; ParamForEvent: LongWord): Integer;
```

Description
Triggers the event OnEnumParOrStyle 127 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);
}
```

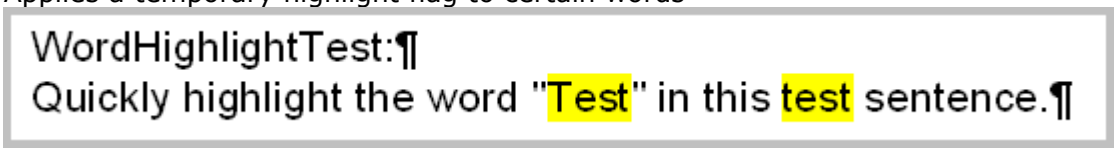
Parameters	
AWord	The word to search for
CaseSensitive	true to work case sensitive
OnlyActiveText	true to only work in current text block
ParamForEvent	Integer value passed through to event

8.5.2.100 IWPTextCursor.WordHighlight

Applies to
IWPTextCursor 219

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

8.5.2.101 IWPTextCursor.SetColumns

Applies to
IWPTextCursor 219

Declaration
```
int SetColumns( int Command,  int Value);
```

Description
Applies properties to control the text column feature. You need the "Premium" license to use this feature.

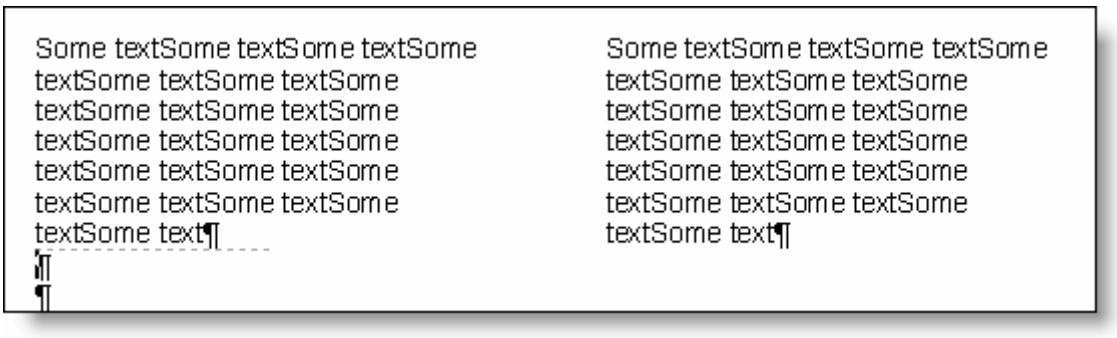
The following command ids are supported:

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();
```



8.5.2.102 IWPTextCursor.SelectCell

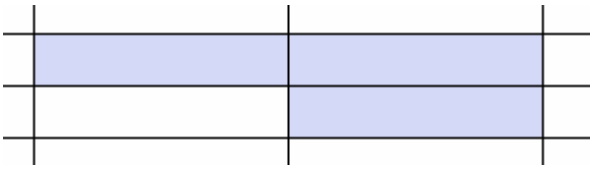
Applies to
IWPTextCursor 219

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 165.

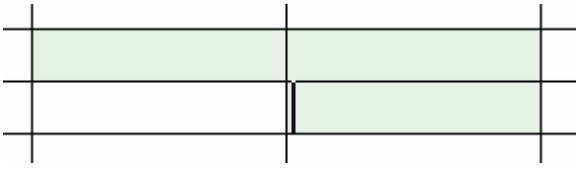
Part 1: Select cells

```
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();
```



8.5.2.103 IWPTextCursor.EnumOpenObj

Applies to
IWPTextCursor 219

Declaration
```
int EnumOpenObj( int ObjType, int EventParam);
```

Description
This method calls the event OnEnumTextObj 127 for all open markers at the current position.

8.5.2.104 IWPTextCursor.ExitTable

Applies to
IWPTextCursor 219

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");
```

8.5.2.105 IWPTextCursor.TableSplit

Applies to
IWPTextCursor 219

Declaration
```
bool TableSplit(bool BeforeRow);
```

Description
Splits the table after or before the current row.

Example:
TextCursor.TableSplit(false);
TextCursor.ExitTable 267();
TextCursor.InputText("Text between tables");
Memo.ReformatAll(false, true);

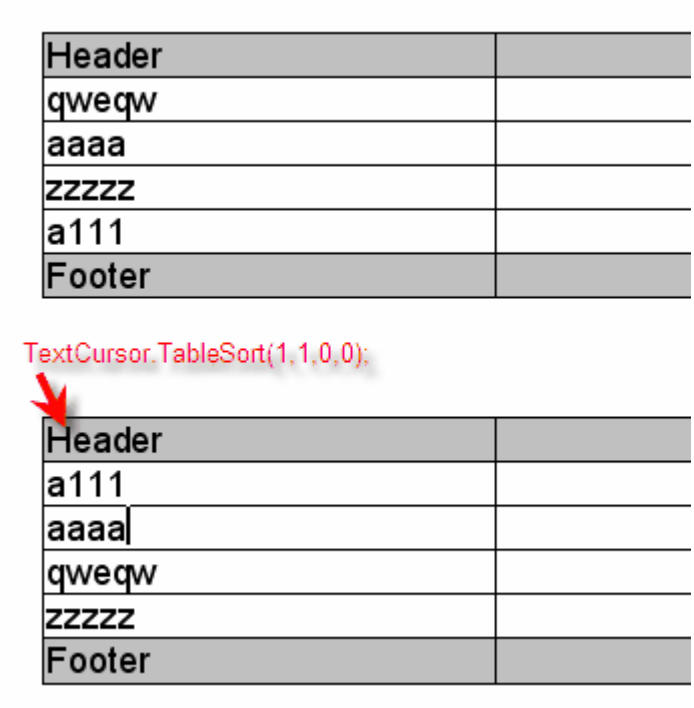
8.5.2.106 IWPTextCursor.TableSort

Applies to
IWPTextCursor 219

Declaration
```
bool TableSort(int HeaderRows, int FooterRow, int SelectColumn, int Method);
```

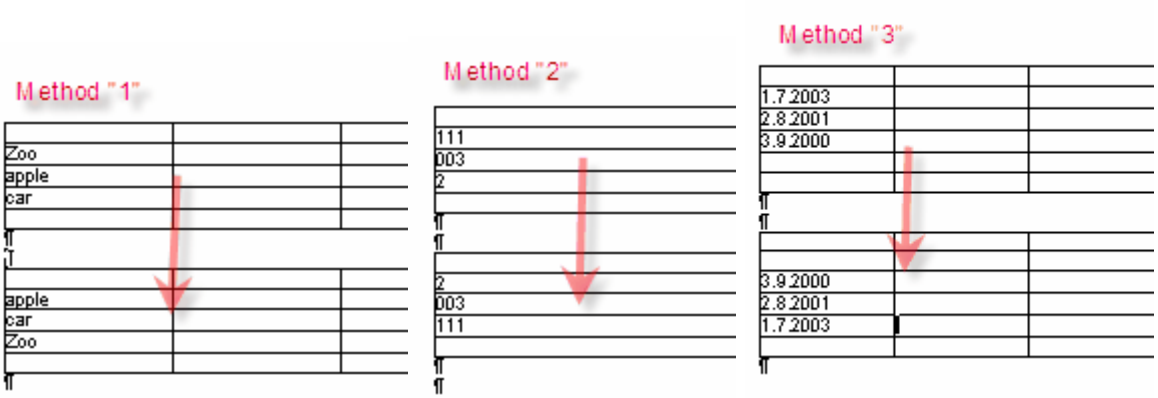
Description
This method sorts the rows in a table.

The given count of header and footer rows are not sorted. You can pass -1 to let header and footer rows be auto detected.



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



8.5.2.107 IWPTextCursor.ASetCellProp

Applies to
IWPTextCursor 219

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.

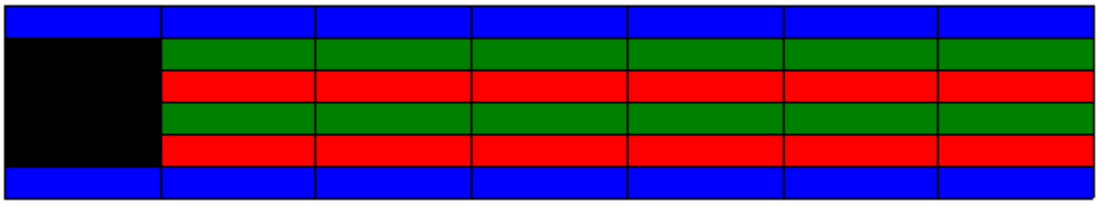
The value of HeaderRows or FooterRows must be the negative to leave this rows unchanged!

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);
```



8.5.2.108 IWPTextCursor.ASetCellStyle

Applies to
IWPTextCursor 219

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!

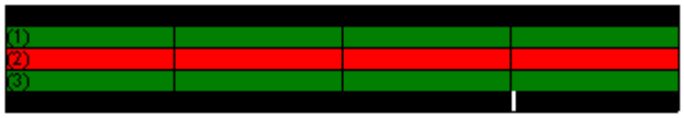
It is possible to only set a range (1...n) of columns or a complete rows with both FromColumn and ToColumn = 0.

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);
```



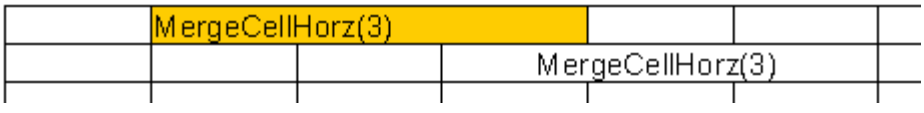
8.5.2.109 IWPTextCursor.MergeCellHorz

Applies to
IWPTextCursor 219

Declaration
```
bool MergeCellHorz(int CellCount);
```

Description
Merges the selected cells with CellCount = 0 (same effect as CombineCellHorz 226).

Otherwise, CellCount must be >1, the number of cells are merged:



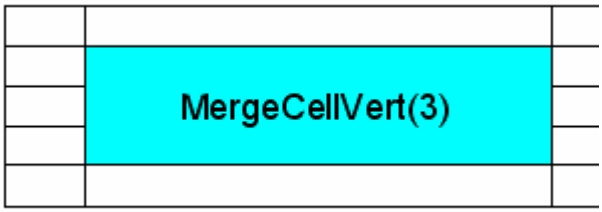
8.5.2.110 IWPTextCursor.MergeCellVert

Applies to
IWPTextCursor 219

Declaration
```
bool MergeCellVert(int CellCount);
```

Description
Merges the selected cells when CellCount = 0. In this case the method works like CombineCellVert 227.

Otherwise the number of Rows (CellCount must be >1) are vertically merged:



8.5.2.111 IWPTextCursor.ClearAllHeaders

Applies to
IWPTextCursor 219

Declaration
```
void ClearAllHeaders();
```

Description
Clears all headers in the text.

8.5.2.112 IWPTextCursor.ClearAllFooters

Applies to
IWPTextCursor 219

Declaration
```
void ClearAllFooters();
```

Description
Clears all footers in the text.

8.6 IWPPdfCreator

Interface to start and configure PDF creation

Description
The integrated PDF engine is based on the powerful wPDF V3 library which is widely accepeted 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.

Properties:

- CIDFontMode 272
- Compression 272
- FontMode 272
- OwnerPW 273
- PDFAMode 273
- PDFFile 273
- Protection 274
- Security 274

Methods:

- BeginDoc 274
- EndDoc 275
- GetProperty 275
- Print 275
- PrintSecond 275
- SetProp 276
- SetProperty 276

UserWP 274

8.6.1 Properties

8.6.1.1 CIDFontMode

Applies to
IWPPdfCreator 271

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, 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

8.6.1.2 Compression

Applies to
IWPPdfCreator 271

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

8.6.1.3 FontMode

Applies to
IWPPdfCreator 271

Declaration

```
int FontMode;
```

Description
Fontmode:
0 : UseTrueTypeFonts
1 : EmbedTrueTypeFonts
2 : EmbedSymbolTrueTypeFonts
3 : UseBase14Type1Fonts
4 : EmbedSubsetTrueType_Charsets
5 : EmbedSubsetTrueType_UsedChar

8.6.1.4 OwnerPW

Applies to
IWPPdfCreator 271

Declaration
```
string OwnerPW;
```

Description
The owner password for the PDF file. Also see UserPW 274.

8.6.1.5 PDFAMode

Applies to
IWPPdfCreator 271

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.

8.6.1.6 PDFFile

Applies to
IWPPdfCreator 271

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.

8.6.1.7 Protection

Applies to
IWPPdfCreator 271

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

8.6.1.8 Security

Applies to
IWPPdfCreator 271

Declaration
```
int Security;
```

Description
Security: 0=40 bit, 1=128 bit

8.6.1.9 UserWP

Applies to
IWPPdfCreator 271

Declaration
```
string UserWP;
```

Description
User Password for PDF file - it will be requested when the PDF file is opened. Also see OwnerPW 273.

Note - actually the name was intended to be UserPW :-)

8.6.2 Methods

8.6.2.1 IWPPdfCreator.BeginDoc

Applies to
IWPPdfCreator 271

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 275 to close the PDF file.

8.6.2.2 IWPPdfCreator.EndDoc

Applies to
IWPPdfCreator 271

Declaration
```
procedure EndDoc;
```

Description
Close a PDF file opened with BeginDoc 274.

8.6.2.3 IWPPdfCreator.GetProperty

Applies to
IWPPdfCreator 271

Declaration
```
function GetProperty(const Name: WideString): WideString;
```

Description
Read a PDF document property, such as "Author", "Keywords", "Subject".

8.6.2.4 IWPPdfCreator.Print

Applies to
IWPPdfCreator 271

Declaration
```
function Print: WordBool;
```

Description
Export the text in editior #1.

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.

Category
Load and Save 150
Printing 156

8.6.2.5 IWPPdfCreator.PrintSecond

Applies to
IWPPdfCreator 271

Declaration
```
function PrintSecond: WordBool;
```

Description

Export the text in editior #2.

8.6.2.6 IWPPdfCreator.SetProp

Applies to
IWPPdfCreator 271

Declaration
```
procedure SetProp(ID: Integer; const Value: WideString);
```

Description
Reserved

8.6.2.7 IWPPdfCreator.SetProperty

Applies to
IWPPdfCreator 271

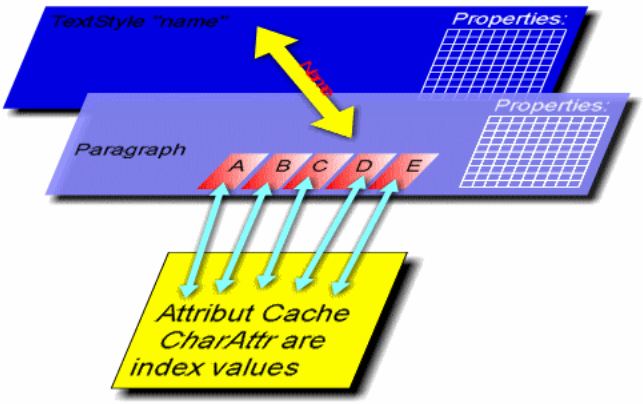
Declaration
```
procedure SetProperty(const Name: WideString; const Value: WideString);
```

Description
Write a PDF document property, such as "Author", "Keywords", "Subject".

8.7 IWPAttrInterface

This important interface is used to update the character attributes and paragraph attributes.

The interface is published by property CurrAttr 162. This instance of the interface changes either the selected text (if text is selected) or the current writing mode. You can use to create your own toolbar, 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 instance "AttrHelper" can be used to calculate CharAttrIndex values to be used with TextCursor. InputString 248.

A different instance is published as property CurrParAttr (Memo.CurrParAttr 164 or RTFDataBlock. CurrParAttr 300). 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 ParASet 362.

CurrStyleAttr 165 is used to change the character attributes defined by a style. You can use SelectStyle 196 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 348.

The interface IWPAttrInterface has several methods to read and write properties, i.e. AttrGet

280 and AttrSet 280. These methods expect the property ID (WPAT...) - Please see the list of the available WPAT 410 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.

The property **CharAttrIndex** 279 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!

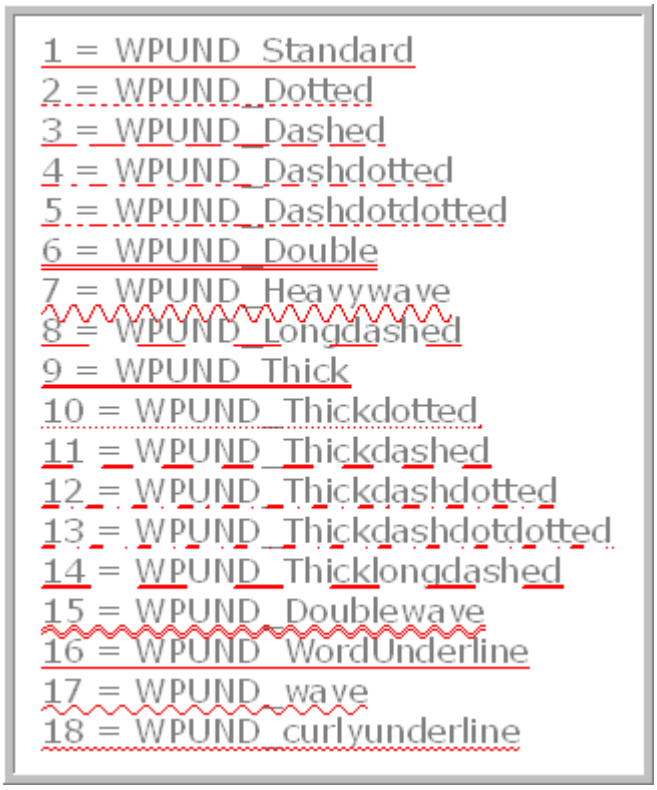
Methods

AttrGet 280
AttrSet 280
BeginUpdate 280
Clear 280
ColorToNr 280
EndUpdate 281
ExcludeStyles 281
GetBGColor 282
GetBGColorNr 282
GetCharStyleSheetName 282
GetCharWidth 282
GetColor 283
GetColorNr 283
GetFontCharset 283
GetFontface 283
GetFontSize 283
GetStyles 284
GetTextLanguage 284
GetUnderlineColor 284
GetUnderlineColorNr 284
GetUnderlineMode 285
GetWPCSS 285

Methods

IncludeStyles 286
NrToColor 286
SetBGColor 287
SetBGColorNr 287
SetCharStyleSheetName 287
SetCharWidth 287
SetColor 288
SetColorNr 288
SetFontCharset 288
SetFontface 288
SetFontSize 289
SetStyles 289
SetTextLanguage 289
SetUnderlineColor 290
SetUnderlineColorNr 290
SetUnderlineMode 290
SetWPCSS 290
ToggleStyle 291

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();
```

8.7.1 Properties

8.7.1.1 CharAttrIndex

Index of the attribute record which holds character properties.

Applies to
IWPAttrInterface 276

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 283 to read the actually used property elements. Methods such as InsertText 358 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 276 can be also used to manipulate the current style 165 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 164 or RTFDataBlock.CurrPar 300). In this case reading CharAttrIndex will return the value which was applied last, it is the "default" attribute for a paragraph .

Category
Character Attributes 146

8.7.2 Methods

8.7.2.1 IWPAttrInterface.AttrDel

Applies to
IWPAttrInterface 276

Declaration
```
procedure AttrDel(WPAT_Code: Integer);
```

Description
Removes a certain property definition, the inherited or the default value will be used for this property.

8.7.2.2 IWPAttrInterface.AttrGet

Applies to
IWPAttrInterface 276

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.

8.7.2.3 IWPAttrInterface.AttrSet

Applies to
IWPAttrInterface 276

Declaration
```
procedure AttrSet(WPAT_Code: Integer; Value: Integer);
```

Description
Sets a certain property.

8.7.2.4 IWPAttrInterface.BeginUpdate

Applies to
IWPAttrInterface 276

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.

8.7.2.5 IWPAttrInterface.Clear

Applies to
IWPAttrInterface 276

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 196 for an example.

8.7.2.6 IWPAttrInterface.ColorToNr

Applies to

IWPAttrInterface 276

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 280 or SetBGColorNr 287 and SetColorNr 288.

Example:

```
Memo.CurrAttr.SetFontSize(8);
Memo.CurrAttr.SetColorNr( Memo.CurrAttr.ColorToNr("red") );
TextCursor.InputText("Some red text");
Memo.CurrAttr.SetColorNr(-1);
```

8.7.2.7 IWPAttrInterface.EndUpdate

Applies to
IWPAttrInterface 276

Declaration
```
function EndUpdate: Boolean;
```

Description
This method has to be used after BeginUpdate 280.

8.7.2.8 IWPAttrInterface.ExcludeStyles

Applies to
IWPAttrInterface 276

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 147

8.7.2.9 IWPAttrInterface.GetBGColor

Applies to
IWPAttrInterface 276

Declaration
```
function GetBGColor(var Color: TColor): Boolean;
```

Description
The method reads the background color used for text as RBG value.

8.7.2.10 IWPAttrInterface.GetBGColorNr

Applies to
IWPAttrInterface 276

Declaration
```
function GetBGColorNr(var ColorNr: Integer): Boolean;
```

Description
The method reads the background color used for text as index value.

8.7.2.11 IWPAttrInterface.GetCharStyleSheetName

Applies to
IWPAttrInterface 276

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 348 which is used to attach a style to a paragraph.

8.7.2.12 IWPAttrInterface.GetCharWidth

Applies to
IWPAttrInterface 276

Declaration
```
function GetCharWidth(var CharWidthPC: Integer): Boolean;
```

Description
If the text uses compressed text, this method will read the width property.

8.7.2.13 IWPAttrInterface.GetColor

Applies to
IWPAttrInterface 276

Declaration
```
function GetColor(var Color: TColor): Boolean;
```

Description
This function reads the current character color.

8.7.2.14 IWPAttrInterface.GetColorNr

Applies to
IWPAttrInterface 276

Declaration
```
function GetColorNr(var ColorNr: Integer): Boolean;
```

Description
This function reads the current character color as index value 412.

8.7.2.15 IWPAttrInterface.GetFontCharset

Applies to
IWPAttrInterface 276

Declaration
```
function GetFontCharset(var Charset: Integer): Boolean;
```

Description
This function reads the charset used by the text.

8.7.2.16 IWPAttrInterface.GetFontface

Applies to
IWPAttrInterface 276

Declaration
```
function GetFontface(var FontName: WideString): Boolean;
```

Description
This method reads the font name used for the text.

8.7.2.17 IWPAttrInterface.GetFontSize

Applies to
IWPAttrInterface 276

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.

8.7.2.18 IWPAttrInterface.GetStyles

Applies to
IWPAttrInterface 276

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 147

8.7.2.19 IWPAttrInterface.GetTextLanguage

Applies to
IWPAttrInterface 276

Declaration
```
function GetTextLanguage(var Mode: Integer): Boolean;
```

Description
This method reads the language id used with this text. It is currently not used.

8.7.2.20 IWPAttrInterface.GetUnderlineColor

Applies to
IWPAttrInterface 276

Declaration
```
function GetUnderlineColor(var RGB: TColor): Boolean;
```

Description
This method reads the color used for underlines.

8.7.2.21 IWPAttrInterface.GetUnderlineColorNr

Applies to

IWPAttrInterface 276

Declaration
```
function GetUnderlineColorNr(var ColorNr: Integer): Boolean;
```

Description
This method reads the color used for underlines as index value.

8.7.2.22 IWPAttrInterface.GetUnderlineMode

Applies to
IWPAttrInterface 276

Declaration
```
function GetUnderlineMode(var Mode: Integer): Boolean;
```

Description
This method reads the mode used for underlines. This are the available values.

```
1 = WPUND_Standard
2 = WPUND_Dotted
3 = WPUND_Dashed
4 = WPUND_Dashdotted
5 = WPUND_Dashdotdotted
6 = WPUND_Double
7 = WPUND_Heavywave
8 = WPUND_Longdashed
9 = WPUND_Thick
10 = WPUND_Thickdotted
11 = WPUND_Thickdashed
12 = WPUND_Thickdashdotted
13 = WPUND_Thickdashdotdotted
14 = WPUND_Thicklongdashed
15 = WPUND_Doublewave
16 = WPUND_WordUnderline
17 = WPUND_wave
18 = WPUND_curlyunderline
```

.

8.7.2.23 IWPAttrInterface.GetWPCSS

Get Definition as string.

Applies to
IWPAttrInterface 276

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 290.

Category
TextDynamic CSS strings 159

8.7.2.24 IWPAttrInterface.IncludeStyles

Applies to
IWPAttrInterface 276

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 147

8.7.2.25 IWPAttrInterface.NrToColor

Applies to
IWPAttrInterface 276

Declaration
```
function NrToColor(NrToColor: Integer): WideString;
```

Description
The engine internally saves colors as index values. This method converts an index into a RGB value.

8.7.2.26 IWPAttrInterface.SetBGColor

Applies to
IWPAttrInterface 276

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)
.
```

8.7.2.27 IWPAttrInterface.SetBGColorNr

Applies to
IWPAttrInterface 276

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 412 used by the RTF engine.

The methods ColorToNr and NrToColor are used to convert between index values and RGB values.

8.7.2.28 IWPAttrInterface.SetCharStyleSheetName

Applies to
IWPAttrInterface 276

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.

8.7.2.29 IWPAttrInterface.SetCharWidth

Applies to
IWPAttrInterface 276

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.

8.7.2.30 IWPAttrInterface.SetColor

Applies to
IWPAttrInterface 276

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)
.
```

8.7.2.31 IWPAttrInterface.SetColorNr

Applies to
IWPAttrInterface 276

Declaration
```
void SetColorNr(int ColorNr);
```

Description
This method is used to set the font color. The parameter is the internal index value 412 used by the RTF engine.

The methods ColorToNr 280 and NrToColor 286 are used to convert between index values and RGB values.

8.7.2.32 IWPAttrInterface.SetFontCharset

Applies to
IWPAttrInterface 276

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.

8.7.2.33 IWPAttrInterface.SetFontface

Applies to
IWPAttrInterface 276

Declaration
```
procedure SetFontface(const FontName: WideString);
```

Description
This method is used to set the name of the font used by the text.

8.7.2.34 IWPAttrInterface.SetFontSize

Applies to
IWPAttrInterface 276

Declaration
```
procedure SetFontSize(Size: Single);
```

Description
Use this method to set the font height in point. The parameter is a floating point value.

8.7.2.35 IWPAttrInterface.SetStyles

Applies to
IWPAttrInterface 276

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 147

8.7.2.36 IWPAttrInterface.SetTextLanguage

Applies to
IWPAttrInterface 276

Declaration
```
procedure SetTextLanguage(Mode: Integer);
```

Description
This method is used to set the language for the text. This method is currently not used.

8.7.2.37 IWPAttrInterface.SetUnderlineColor

Applies to
IWPAttrInterface 276

Declaration
```
procedure SetUnderlineColor(Color: TColor);
```

Description
This method sets the color for the underline as RGB value. See example "Underline Modes".

8.7.2.38 IWPAttrInterface.SetUnderlineColorNr

Applies to
IWPAttrInterface 276

Declaration
```
procedure SetUnderlineColorNr(ColorNr: Integer);
```

Description
This method sets the color for the underline as color index value.

8.7.2.39 IWPAttrInterface.SetUnderlineMode

Applies to
IWPAttrInterface 276

Declaration
```
procedure SetUnderlineMode(Mode: Integer);
```

Description
This method sets the mode used for underlines. See GetUnderlineModes.

8.7.2.40 IWPAttrInterface.SetWPCSS

Applies to
IWPAttrInterface 276

Declaration
```
procedure SetWPCSS(const wpcss: WideString);
```

Description
Set the attributes saved to a string using GetWPCSS 285.

Category
TextDynamic CSS strings 159

8.7.2.41 IWPAttrInterface.ToggleStyle

Applies to
IWPAttrInterface 276

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 160
  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);
     }
}
```

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)

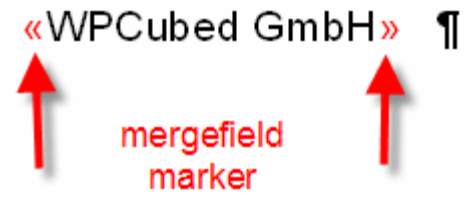
Category
Character Styles 147

8.8 IWPCharacterAttr

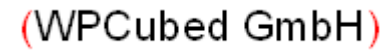
Attributes for links and fields

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**,"')");



It is also possible to display the name in one of the markers using the string "%N(":

(%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 293	Italic 294	SetCodeText 296
Bold 293	StrikeOut 295	
CodeTextColor 293	SubScript 295	
DoubleUnderline 294	SuperScript 295	
Hidden 294	TextColor 295	
HotEffect 294	Underline 295	
HotTextColor 294	UnderlineColor 296	

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

8.8.1 Properties

8.8.1.1 BackgroundColor

Applies to
IWPCharacterAttr 292

Declaration
```
int BackgroundColor;
```

Description
The background color as RGB value, can be used for merged text, bookmarked text and hyperlinks.

8.8.1.2 Bold

Applies to
IWPCharacterAttr 292

Declaration
```
ThreeState Bold;
```

Description
Use a bold font, can be used for merged text, bookmarked text and hyperlinks.

8.8.1.3 CodeTextColor

Applies to
IWPCharacterAttr 292

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.

8.8.1.4 DoubleUnderline

Applies to
IWPCharacterAttr 292

Declaration
```
property DoubleUnderline: WordBool read Get_DoubleUnderline write
Set_DoubleUnderline;
```

Description
Use double underlining, can be used for merged text, bookmarked text and hyperlinks.

8.8.1.5 Hidden

Applies to
IWPCharacterAttr 292

Declaration
```
bool Hidden;
```

Description
Hide the code objects

8.8.1.6 HotEffect

Applies to
IWPCharacterAttr 292

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)

8.8.1.7 HotTextColor

Applies to
IWPCharacterAttr 292

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)

8.8.1.8 Italic

Applies to
IWPCharacterAttr 292

Declaration
```
ThreeState Italic;
```

Description

Use an italic font, can be used for merged text, bookmarked text and hyperlinks.

8.8.1.9 StrikeOut

Applies to
IWPCharacterAttr 292

Declaration
```
ThreeState StrikeOut;
```

Description
```
Use tsTrue (value=1) to strike out the text.
```

8.8.1.10 SubScript

Applies to
IWPCharacterAttr 292

Declaration
```
ThreeState SubScript;
```

Description
Use a subscript font.

8.8.1.11 SuperScript

Applies to
IWPCharacterAttr 292

Declaration
```
ThreeState SuperScript;
```

Description
Use a superscript font.

8.8.1.12 TextColor

Applies to
IWPCharacterAttr 292

Declaration
```
ThreeState TextColor;
```

Description
The color for the embedded text. This does not change the color for the objects.

8.8.1.13 Underline

Applies to
IWPCharacterAttr 292

Declaration
```
ThreeState Underline;
```

Description

Use underlining.

8.8.1.14 UnderlineColor

Applies to
IWPCharacterAttr 292

Declaration
```
int UnderlineColor;
```

Description
Change the color for the underline.

8.8.2 Methods

8.8.2.1 IWPCharacterAttr.SetCodeText

Applies to
IWPCharacterAttr 292

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
```

8.9 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 162 and also Memo.BlockAdd 176, Memo.BlockAppend 177 and Memo.BlockFind 178.

An interface to this element can be useful to create header or footer in code. You can use the method Clear 302 to erase the text. Use Delete 302 to remove the layer completely.

You can append one paragraph using AppendParagraph 301. This can be useful to modify the text without having to move the cursor. Alternatively, to select the first paragraph with SelectFirstPar 303. In both cases the interfaces CurrPar 300 and CurrParAttr 300 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 368 to locate a different paragraph! (Make sure to call ReformatAll 191 after the text has been updated.)

To move the cursor into the layer set property WorkOnText 300 to true.

The property Kind 298 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
}
```

Properties

ID 297
IsEmpty 297
Kind 298
Name 298
Range 299
Readonly 299
SectionID 299
Text 299
WorkOnText 300
CurrPar 300
CurrParAttr 300

Methods

AppendParagraph 301
Clear 302
Delete 302
GetParPtrFirst 302
GetParPtrLast 302
SelectFirstPar 303

8.9.1 Properties

8.9.1.1 ID

Applies to
IWPDataBlock 296

Declaration
```
int ID;
```

Description
This is the ID of the data block. IDs are used to select certain data blocks in event OnGetSpecialText.

8.9.1.2 IsEmpty

Applies to
IWPDataBlock 296

Declaration
```
bool IsEmpty;
```

Description
This property is true if the data block is completely empty.

8.9.1.3 Kind

Applies to
IWPDataBlock 296

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.

8.9.1.4 Name

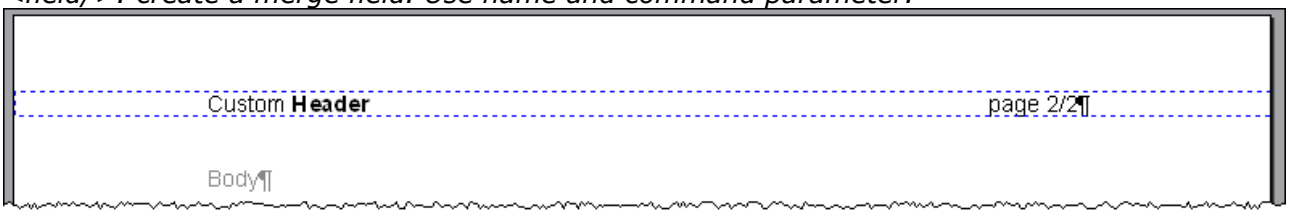
Applies to
IWPDataBlock 296

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
     );
     }
}
```

8.9.1.5 Range

Applies to
IWPDataBlock 296

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.

8.9.1.6 Readonly

Applies to
IWPDataBlock 296

Declaration
```
bool Readonly;
```

Description
If this property is true the layer cannot be edited.

8.9.1.7 SectionID

Applies to
IWPDataBlock 296

Declaration
```
int SectionID;
```

Description
This id is used to select a header or footer for a certain section in the text.

8.9.1.8 Text

Applies to
IWPDataBlock 296

Declaration
```
string Text;
```

Description
The property Text can be used to assign new text. Example see property name 298.

This property is write - only!

Category
Load and Save 150

8.9.1.9 WorkOnText

Applies to
IWPDataBlock 296

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.

8.9.1.10 CurrPar

Applies to
IWPDataBlock 296

Declaration
IWPParInterface 339 `ParAttr`

Description

This interface let you modify the current paragraph. The current paragraph is the paragraph which was appended last using the function AppendParagraph 301. The method SelectFirstPar 303 will make the first paragraph the "current".

Note: Please don't forget to call ReleaseInt() 123 with the returned interface at the end of your code.

8.9.1.11 CurrParAttr

Applies to
IWPDataBlock 296

Declaration
IWPAttrInterface 276 `ParAttr`

Description

This interface let you modify the current paragraph. The current paragraph is the paragraph which was appended last using the function AppendParagraph 301. The method SelectFirstPar 303 will make the first paragraph the "current".

Note: Please don't forget to call ReleaseInt() 123 with the returned interface at the end of your code.

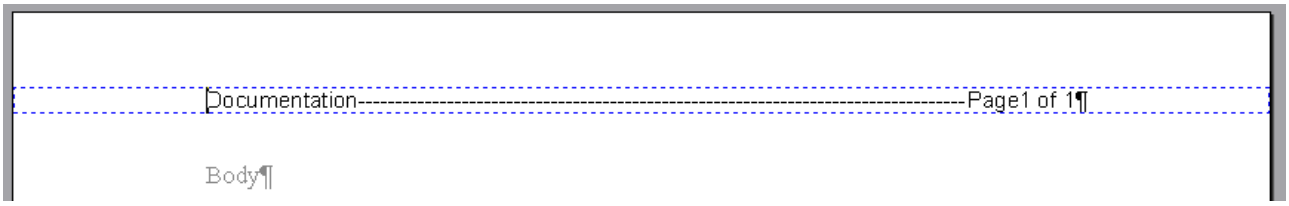
8.9.2 Methods

8.9.2.1 IWPDataBlock.AppendParagraph

Applies to
IWPDataBlock 296

Declaration
IWPParInterface 339 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 366. 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 158

8.9.2.2 IWPDataBlock.Clear

Applies to
IWPDataBlock 296

Declaration
```
void Clear();
```

Description
This procedure clears the text in this block.

8.9.2.3 IWPDataBlock.Delete

Applies to
IWPDataBlock 296

Declaration
```
void Delete();
```

Description
To delete the entire block use the procedure Delete.

8.9.2.4 IWPDataBlock.GetParPtrFirst

Applies to
IWPDataBlock 296

Declaration
```
int GetParPtrFirst()
```

Description
This is the ID of the first paragraph in this text block.

8.9.2.5 IWPDataBlock.GetParPtrLast

Applies to
IWPDataBlock 296

Declaration
```
int GetParPtrFirst()
```

Description
This is the ID of the last paragraph in this text block.

8.9.2.6 IWPDataBlock.SelectFirstPar

Applies to
IWPDataBlock 296

Declaration
```
bool SelectFirstPar()
```

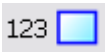
Description
This method selects the the first paragraph in this text block to be the "current". You can then use CurrPar 300 and CurrParAttr 300 to manipulate this paragraph. To move to a different paragraph use, for example, CurrPar.SelectNextPar().

If the text block is empty the function will return false. You can use AppendParagraph 301 to create a new paragraph.

8.10 IWPDllButton

Read properties fo custom buttons

Description
The interface IWPDllButton is mainly used by the event OnButtonClick which is triggered when the user clicks on a button on any of the TextDynamic toolpanels. The properties defined by IWPDllButton let you read the values assigned by the XML description of the toolbar.



was created by this XML code in the EditLayout XML script.

```
<ToolBar ShowCaption="0">
  <!-- custom buttons -->
  <Button image=147 Name="Test" Param="Something" Caption="123" ShowCaption=1/>
```

You can read the property Name in the OnButtonClick 135 event to create a new action.

Properties
- **Action** 304
- **Caption** 304
- **Disabled** 304
- **Font** 304
- **Hint** 304
- **Image** 305
- **IParam** 305
- **Name** 305
- **param** 305
- **Selected** 305
- **ShowCaption** 306
- **Typ** 306
- **Visible** 306
- **WPActionStyle** 307

Methods
- **ItemsAdd** 307
- **ItemsClear** 307

8.10.1 Properties

8.10.1.1 Action

Applies to
IWPDllButton 303

Declaration
```
string Action;
```

Description
This string is set by XML parameter "Action".

It it is also used to link a menu or button to a custom action 418.

8.10.1.2 Caption

Applies to
IWPDllButton 303

Declaration
```
string Caption;
```

Description
This string is set by XML parameter "Caption". It will only be displayed if property ShowCaption of the containing toolpanel or the button itself has been activated.

8.10.1.3 Disabled

Applies to
IWPDllButton 303

Declaration
```
bool Disabled;
```

Description
If this property is true the alternativem shaded image will be displayed.

8.10.1.4 Font

Applies to
IWPDllButton 303

Declaration
```
string Font;
```

Description
This font will be used to draw the caption.

8.10.1.5 Hint

Applies to
IWPDllButton 303

Declaration

```
string Hint;
```

Description
This hint string will be sent to event OnShowHint 142 to used to create the fly over hint window.

8.10.1.6 Image

Applies to
IWPDllButton 303

Declaration
```
int Image;
```

Description
An image index used to locate the correct image in the image list.

8.10.1.7 IParam

Applies to
IWPDllButton 303

Declaration
```
int IParam;
```

Description
An integer parameter also read from XML.

8.10.1.8 Name

Applies to
IWPDllButton 303

Declaration
```
string Name;
```

Description
This string is set by XML parameter "Name".

8.10.1.9 param

Applies to
IWPDllButton 303

Declaration
```
string param;
```

Description
This string is set by XML parameter "Param".

8.10.1.10 Selected

Applies to
IWPDllButton 303

Declaration

```
bool Selected;
```

Description
If true the button will be displayed highlighted or pressed.

8.10.1.11 ShowCaption

Applies to
IWPDllButton 303

Declaration
```
bool ShowCaption;
```

Description
If this property is true the caption will be displayed.

8.10.1.12 Typ

Applies to
IWPDllButton 303

Declaration
```
int Typ;
```

Description
The following "button" types are supported:

Value	Typ	XML tag	Meaning
0	wpbButton	<button/>	Button, can be pressed but not selected
1	wpbCollection	<collection/>	Like the button but with contained controls. It inherits the Image of the first element inside this collection which is selected!
2	wpbCheck	<check/>	Button which can be selected
3	wpbRadio	<radio/>	Like "check" but when selected all in same group are de-selected
4	wpbSubMenu	<submenu/>	Button - will show its sub menu if clicked
5	wpbDropDown	<dropdown/>	Button with drop down combobox
6	wpbDropDownList	<dropdownlist/>	Combobox, it can be edited
7	wpbSeparator	<separator/>	Vertical or horizontal line
8	wpbInfoText	<text/>	Some text, cannot be clicked
9	wpbTab	<tab/>	Like 'Radio' but with angled sides. (Used by PanelH2 only!)

8.10.1.13 Visible

Applies to
IWPDllButton 303

Declaration
```
bool Visible;
```

Description
If this property is false the button will nbot be displayed.

8.10.1.14 WPActionStyle

Applies to
IWPDllButton 303

Declaration
```
int WPActionStyle;
```

Description
This is the integer ID of the WPA action style. Please note that this is not a fixed number, it may change with an update.

8.10.2 Methods

8.10.2.1 IWPDllButton.ItemsAdd

Applies to
IWPDllButton 303

Declaration
```
procedure ItemsAdd(const Text: WideString; param: Integer);
```

Description
This method can be used to add text items to a dowp down list.

8.10.2.2 IWPDllButton.ItemsClear

Applies to
IWPDllButton 303

Declaration
```
procedure ItemsClear;
```

Description
This method clears all items in a dowp down list.

8.11 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 309 - this also works for formatted text.

You can use the method ContinueOptions 311 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 314 and LoadPicture 314. 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 313. You can also load a file with

formatted text using LoadText 314

To create a table with data inside the field execute AddTable 310. 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 312. The field markers, not the contents will be removed.

An interface to access the current field is provided as property FieldObject 313. You can use this interface to read other properties of the merge field.

Properties

Description 308
FieldCommand 308
FieldName 309
FloatValue 309
Format 309
IsMergefield 309
StringValue 309
Title 310

Methods

AddTable 310
ContinueOptions 311
CurrentBand 312
CurrentGroup 312
DeleteField 312
EmbeddedObject 312
ExecStrCommand 313
FieldAttr 313
FieldObject 313
InputHyperlink 313
LoadImage 314
LoadPicture 314
LoadText 314
SetValue 314

8.11.1 Properties

8.11.1.1 Description

Applies to
IWPFieldContents 307

Declaration
```
string Description;
```

Description

Only used during report creation to red predefined value.

8.11.1.2 FieldCommand

Applies to
IWPFieldContents 307

Declaration
```
string FieldCommand;
```

Description
The property gives access to the "command" property of the field.

8.11.1.3 FieldName

Applies to
IWPFieldContents 307

Declaration
```
string FieldName;
```

Description
The property gives access to the name of the field.

8.11.1.4 FloatValue

Applies to
IWPFieldContents 307

Declaration
```
double FloatValue;
```

8.11.1.5 Format

Applies to
IWPFieldContents 307

Declaration
```
string Format;
```

Description

Only used during report creation to red predefined value.

8.11.1.6 IsMergefield

Applies to
IWPFieldContents 307

Declaration
```
bool IsMergefield;
```

8.11.1.7 StringValue

Applies to
IWPFieldContents 307

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 189 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 313 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 188. The input methods of the IWPTextCursor 219 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 154

8.11.1.8 Title

Applies to
IWPFieldContents 307

Declaration
```
string Title;
```

Description

Only used during report creation to red predefined value.

8.11.2 Methods

8.11.2.1 IWPFieldContents.AddTable

Applies to
IWPFieldContents 307

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 125 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 148
Table Support 158

8.11.2.2 IWPFieldContents.ContinueOptions

Applies to
IWPFieldContents 307

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 154

8.11.2.3 IWPFieldContents.CurrentBand

Applies to
IWPFieldContents 307

Declaration
```
function CurrentBand: IWPReportBand 376;
```

8.11.2.4 IWPFieldContents.CurrentGroup

Applies to
IWPFieldContents 307

Declaration
```
function CurrentGroup: IWPReportBand 376;
```

8.11.2.5 IWPFieldContents.DeleteField

Applies to
IWPFieldContents 307

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 310) 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 309 is inserted.

8.11.2.6 IWPFieldContents.EmbeddedObject

Applies to
IWPFieldContents 307

Declaration
```
function EmbeddedObject: IWPTextObj 396;
```

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.

8.11.2.7 IWPFieldContents.ExecStrCommand

Applies to
IWPFieldContents 307

Declaration
```
procedure ExecStrCommand(CommandID: Integer; param: Integer; const StrParam:
WideString);
```

Description
This method is reserved for custom manipulations of the inserted text - if you have a wish please let us know.

8.11.2.8 IWPFieldContents.FieldAttr

Applies to
IWPFieldContents 307

Declaration
```
function FieldAttr: IWPAttrInterface 276;
```

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 146

8.11.2.9 IWPFieldContents.FieldObject

Applies to
IWPFieldContents 307

Declaration
```
function FieldObject: IWPTextObj 396;
```

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.

8.11.2.10 IWPFieldContents.InputHyperlink

Applies to
IWPFieldContents 307

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 149

8.11.2.11 IWPFieldContents.LoadImage

Applies to
IWPFieldContents 307

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.

8.11.2.12 IWPFieldContents.LoadPicture

Applies to
IWPFieldContents 307

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 );
```

8.11.2.13 IWPFieldContents.LoadText

Applies to
IWPFieldContents 307

Declaration
```
procedure LoadText(const filename: WideString);
```

Description
This method can be used to load a file which will be inserted.

Category
Load and Save 150

8.11.2.14 IWPFieldContents.SetValue

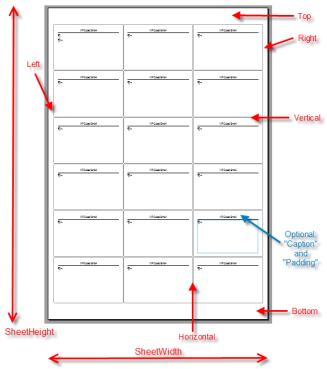
Applies to
IWPFieldContents 307

Declaration
```
procedure SetValue(Value: OleVariant);
```

8.12 IWPLabelDef

Description

Using the interface LabelDef 166 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 label number to start with. All parameters of a label can be specified, using centimeter or inch values, depending on UnitIsInch 319.



Also see "Mail Merge 76" and IWPPageSizeList 338.

A Label is either defined by the sheet size (width/ height), column count, rowcount and the margins (top, bottom,left, right, horizontal and vertical)
or, alternatively
by the sheet size (width/ height), the top, left, right and bottom margin and the specified label width and label height.

Active 315 : Switch Label display on / off
AsText 316 : Retrieve and set the label definition as text
Bottom 316 : Bottom Margin
Caption 316 : Caption of the label, displayed over text
ColumnCount 316 : Count of columns
Horizontal 317 : Horizontal margin between columns
LabelHeight 317 : Size of a label, if you set it you cannot change vertical margin and row count
LabelWidth 317 : Size of a label, if you set it you cannot change horizontal margin and row count
Left 317 : Left Margin
Name 317 : Name of this label definition (encoded into "AsText")
Padding 318 : Padding inside of the label
Right 318 : Right Margin
RowCount 318 : Count of rows
SheetHeight 318 : Height of label sheet
SheetWidth 318 : Width of label sheet
StartNr 319 : Start nr for printing
Top 319 : Top Margin
UnitIsInch 319 : If true all floating point values are inch instead of CM
Vertical 319 : Margin between rows

8.12.1 Properties

8.12.1.1 Active

Applies to
IWPLabelDef 314

Declaration
```
bool Active;
```

Description
When this property is set to **true**, the editor will display a label sheet. The page size which is currently defined in the editor is overridden when this mode is on. The event OnMeasurePage is disabled.

8.12.1.2 AsText

Applies to
IWPLabelDef 314

Declaration
```
string AsText;
```

Description
Retrieves or sets all parameters except for Caption, Active and StartNr as string. This methods makes it easy to load and save predefined label definitions.

8.12.1.3 Bottom

Applies to
IWPLabelDef 314

Declaration
```
float Bottom;
```

Description
The height of the bottom margin in cm or inch.

8.12.1.4 Caption

Applies to
IWPLabelDef 314

Declaration
```
string Caption;
```

Description
An optional caption for each label. This caption is printed on each label in a small font (Arial Narrow). You can use it to specify the return address.

8.12.1.5 ColumnCount

Applies to
IWPLabelDef 314

Declaration
```
int ColumnCount;
```

Description
The count of label columns on the sheet.

8.12.1.6 Horizontal

Applies to
IWPLabelDef 314

Declaration
```
float Horizontal;
```

Description
The horizontal margin between labels (in CM or inch).

8.12.1.7 LabelHeight

Applies to
IWPLabelDef 314

Declaration
```
float LabelHeight;
```

Description
The height of the label. You can read this property to display the current height. If written, the properties RowCount and Vertical are changed accordingly.

8.12.1.8 LabelWidth

Applies to
IWPLabelDef 314

Declaration
```
float LabelWidth;
```

Description
The width of the label. You can read this property to display the current width. If written, the properties ColumnCount and Horizontal are changed accordingly.

8.12.1.9 Left

Applies to
IWPLabelDef 314

Declaration
```
float Left;
```

Description
The margin on the left of the label sheet.

8.12.1.10 Name

Applies to
IWPLabelDef 314

Declaration
```
string Name;
```

Description

The name of the label. Will be saved with property AsText.

8.12.1.11 Padding

Applies to
IWPLabelDef 314

Declaration
```
float Padding;
```

Description
The margin between text and label borders on all sides of the label. If this property is 0, the labels will be displayed in the preview as simple rectangles, if it is >0 a round rectangle will be drawn.

8.12.1.12 Right

Applies to
IWPLabelDef 314

Declaration
```
float Right;
```

Description
The margin to the right of the label sheet.

8.12.1.13 RowCount

Applies to
IWPLabelDef 314

Declaration
```
int RowCount;
```

Description
The count of labe rows.

8.12.1.14 SheetHeight

Applies to
IWPLabelDef 314

Declaration
```
float SheetHeight;
```

Description
The height of the label sheet (cm or Inch, depending on UnitIsInch 319).

8.12.1.15 SheetWidth

Applies to
IWPLabelDef 314

Declaration
```
float SheetWidth;
```

Description
The width of the label sheet (cm or Inch, depending on UnitIsInch 319).

8.12.1.16 StartNr

Applies to
IWPLabelDef 314

Declaration
```
int StartNr;
```

Description
The number of the label on first page to start with. This property is useful if you work with half full label sheets.

8.12.1.17 Top

Applies to
IWPLabelDef 314

Declaration
```
float Top;
```

Description
The margin at the top of the label sheet.

8.12.1.18 UnitIsInch

Applies to
IWPLabelDef 314

Declaration
```
bool UnitIsInch;
```

Description
If this value is true all floating point properties are interpreted as **inch** values, otherwise **CM** are expected.

8.12.1.19 Vertical

Applies to
IWPLabelDef 314

Declaration
```
float Vertical;
```

Description
The vertical margin between labels (in CM or inch).

8.13 IWPMapi

Interface to create and send e-mails

Description

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 325) 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 321.

When you execute Prepare 325 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 formated text, first assign the string "<html>" to the property Body 322 .

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 325 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 324 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 to the e-mail client. (Internally the Windows API MAPISendMail is used. The parameter lhSession is passed as 0, ulUIParam can be set using the method SetAppHandle, lpMessage will be automatically prepared and flFlags can be set using the property MAPIFlags)

Send2: Send the e-mail which was prepared last. If no e-mail was prepared the document from editor 2 will be send to the e-mail client.

Clear: Resets the properties Subject 323, FromName 322, FromAddress 322, Recipients 323, CCList 322, BCCList 322, AttachmentList 321, Body 322.

```
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();
}
```

Note

There is another possibility to create an e-mail. You can use the integrated **MIME** writer class.

If you use the method SaveToString 195 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 beeing created! The e-mail fields "From", "To", "CC", "BCC" and "Subject" will be use the value from the respective properties in the MAPI interface.

The MIME writer uses functions of the free library Synapse at http://www.ararat.cz/synapse/

Properties

Methods

8.13.1 Properties

8.13.1.1 AddHTML

Applies to
IWPMapi 319

Declaration
```
bool AddHTML;
```

Description
If true add HTML text as attachment - unless the body is not HTML.

8.13.1.2 AddPDF

Applies to
IWPMapi 319

Declaration
```
bool AddPDF;
```

Description
Reserved.

8.13.1.3 AttachmentList

Applies to
IWPMapi 319

Declaration
```
string AttachmentList;
```

Description
List of attached files. You can read this list after Prepare 325 was used to know which files have to be added to the e-mail.

8.13.1.4 BCCList

Applies to
IWPMapi 319

Declaration
```
string BCCList;
```

Description
e-mail property BCC

8.13.1.5 Body

Applies to
IWPMapi 319

Declaration
```
string Body;
```

Description
The body text of the e-mail. If empty Prepare will fill it with the ANSI version of the document, if it is %gt;html< the HTML code will be used as body.

8.13.1.6 CCList

Applies to
IWPMapi 319

Declaration
```
string CCList;
```

Description
e-mail property CC

8.13.1.7 FromAddress

Applies to
IWPMapi 319

Declaration
```
string FromAddress;
```

Description
e-mail of sender

8.13.1.8 FromName

Applies to
IWPMapi 319

Declaration
```
string FromName;
```

Description
name of sender

8.13.1.9 MAPIFlags

Applies to
IWPMapi 319

Declaration
```
int MAPIFlags;
```

Description
Flags for the MapiSendMail function.

8.13.1.10 Recipients

Applies to
IWPMapi 319

Declaration
```
string Recipients;
```

Description
The recipients for the e-mail.

8.13.1.11 Subject

Applies to
IWPMapi 319

Declaration
```
string Subject;
```

Description
The subject of the e-mail.

8.13.2 Methods

8.13.2.1 IWPMapi.AddBCCRecipient

Applies to
IWPMapi 319

Declaration
```
procedure AddBCCRecipient(const Name: WideString; const EMAIL: WideString);
```

Description
Add e-mail blind copy recipient. Internally the syntax name "<email>" is used.

8.13.2.2 IWPMapi.AddCCRecipient

Applies to
IWPMapi 319

Declaration
```
procedure AddCCRecipient(const Name: WideString; const EMAIL: WideString);
```

Description
Add e-mail copy recipient. Internally the syntax name "<email>" is used.

8.13.2.3 IWPMapi.AddRecipient

Applies to
IWPMapi 319

Declaration
```
procedure AddRecipient(const Name: WideString; const EMAIL: WideString);
```

Description
Add e-mail (or copy) recipient. Internally the syntax name "<email>" is used.

8.13.2.4 IWPMapi.AppendFile

Applies to
IWPMapi 319

Declaration
```
procedure AppendFile(const filename: WideString; const DisplayName: WideString);
```

Description
Add an attachment. You need to use this method after using Prepare 325 since it will clear the attachment list.

8.13.2.5 IWPMapi.Clear

Applies to
IWPMapi 319

Declaration
```
procedure Clear;
```

Description
Resets the e-mail properties to default.

8.13.2.6 IWPMapi.Command

Applies to
IWPMapi 319

Declaration
```
function Command(ID: Integer; const param: WideString; const Param2:
WideString): Integer;
```

Description

You can use ID=1 to delete all temporary files. Otherwise they will deleted at shut down of the application.

8.13.2.7 IWPMapi.InitEmailDLL

Applies to
IWPMapi 319

Declaration
```
function InitEmailDLL(const DLLName: WideString; const param: WideString):
Integer;
```

Description
Reserved to initialize a e-mail server tool.

8.13.2.8 IWPMapi.Prepare

Applies to
IWPMapi 319

Declaration
```
procedure Prepare(Editor: Integer; Options: Integer);
```

Description
Initializes the e-mail. Clears the list of attached files and recreates the Body (if empty) and add the HTML message and images.

8.13.2.9 IWPMapi.Send

Applies to
IWPMapi 319

Declaration
```
function Send: Integer;
```

Description
Send the prepared e-mail using MapiSendMail. If no e-mail was prepared the it will prepare an e-mail using the text in editor #1 and after that clear all temporary files.

Returns
The result value of MapiSendMail().

8.13.2.10 IWPMapi.Send2

Applies to
IWPMapi 319

Declaration
```
function Send2: Integer;
```

Description
Send the prepared e-mail using MapiSendMail. If no e-mail was prepared the it will prepare an e-mail using the text in editor #2 and after that clear all temporary files.

Returns

The result value of MapiSendMail().

8.13.2.11 IWPMapi.SetAppHandle

Applies to
IWPMapi 319

Declaration
```
procedure SetAppHandle(aHandle: Integer);
```

Description
Set the application handle, this value is used for parameter #2 of MapiSendMail.

8.13.2.12 IWPMapi.SetTEMPDir

Applies to
IWPMapi 319

Declaration
```
procedure SetTEMPDir(const Dir: WideString);
```

Description
Set the temporary directory which is used to create HTML and image files.

8.14 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 326	**MarginLeft** 327
ColCount 327	**MarginRight** 327
Height 327	**MarginTop** 328
MarginBottom 327	**PageNr** 328
	Width 328

8.14.1 Properties

8.14.1.1 Changed

Applies to
IWPMeasurePageParam 326

Declaration
```
bool Changed;
```

Description
Assign true to activated the changes.

8.14.1.2 ColCount

Applies to
IWPMeasurePageParam 326

Declaration
```
int ColCount;
```

Description
Reserved.

8.14.1.3 Height

Applies to
IWPMeasurePageParam 326

Declaration
```
int Height;
```

Description
Height of the page in twips (inch/1440)

8.14.1.4 MarginBottom

Applies to
IWPMeasurePageParam 326

Declaration
```
int MarginBottom;
```

Description
Bottom margin in twips (inch/1440)

8.14.1.5 MarginLeft

Applies to
IWPMeasurePageParam 326

Declaration
```
int MarginLeft;
```

Description
Left margin in twips (inch/1440)

8.14.1.6 MarginRight

Applies to
IWPMeasurePageParam 326

Declaration
```
int MarginRight;
```

Description
Right margin in twips (inch/1440)

8.14.1.7 MarginTop

Applies to
IWPMeasurePageParam 326

Declaration
```
int MarginTop;
```

Description
Top margin in twips (inch/1440)

8.14.1.8 PageNr

Applies to
IWPMeasurePageParam 326

Declaration
```
int PageNr;
```

Description
The page number of the page.

8.14.1.9 Width

Applies to
IWPMeasurePageParam 326

Declaration
```
int Width;
```

Description
The width of the page in twips (inch/1440)

8.15 IWPNumberStyle

Interface to modify a numbering style.

Description
This interface is provided by GetNumberStyle 183. 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
 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,
"");
```

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.TextA= "."; // after text =
"."
  style.TextB= "";  // before text
= ""
  style.Indent = 720; // 1/2 inch
indent
  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;
```

```
par.SetText("continued
C",0);
par.NumberLevel = 3;
cursor.InputParagraph(0,
"");
par.SetText("not
numbered",0);
par.NumberLevel = 0;
cursor.InputParagraph(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);
```

Properties

Methods

8.15.1 Properties

8.15.1.1 Color

Applies to
IWPNumberStyle 328

Declaration
`int Color;`

Description
The color for the number text (or bullets) as RGB value.

8.15.1.2 Font

Applies to
IWPNumberStyle 328

Declaration
```
string Font;
```

Description
The font for numbering or bullets. If empty use the paragraph phont.

8.15.1.3 Group

Applies to
IWPNumberStyle 328

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 331 can be defined in an outline style group.

8.15.1.4 ID

Applies to
IWPNumberStyle 328

Declaration
```
int ID;
```

Description
The ID used for the paragraph property WPAT_NumberStyle 414.

8.15.1.5 Indent

Applies to
IWPNumberStyle 328

Declaration
```
int Indent;
```

Description
The width reserved for the number or the bullet.

8.15.1.6 LegalNumbering

Applies to
IWPNumberStyle 328

Declaration
```
bool LegalNumbering;
```

Description
If true use legal numbering: 1.1.1

8.15.1.7 Level

Applies to
IWPNumberStyle 328

Declaration
```
int Level;
```

Description
0 or the level in an outline group 330 between 1 and 9.

8.15.1.8 Mode

Applies to
IWPNumberStyle 328

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

8.15.1.9 Size

Applies to
IWPNumberStyle 328

Declaration
```
float Size;
```

Description
The font size for the numbering, 0 to use paragraph attribute.

8.15.1.10 TextA

Applies to
IWPNumberStyle 328

Declaration
```
string TextA;
```

Description
The text **after** the number.

8.15.1.11 TextB

Applies to

IWPNumberStyle 328

Declaration
```
string TextB;
```

Description
The text **before** the number. You can use it to define the bullet symbol. For bullets also change Font 329.

8.15.1.12 WPCSS

Applies to
IWPNumberStyle 328

Declaration
```
string WPCSS;
```

Description
Get and set the properies as WPCSS string.

8.15.2 Methods

8.15.2.1 IWPNumberStyle.ADel

Applies to
IWPNumberStyle 328

Declaration
```
procedure ADel(WPAT_Code: Integer);
```

Description
Low level method to delete a property referenced by a WPAT id.

8.15.2.2 IWPNumberStyle.AGet

Applies to
IWPNumberStyle 328

Declaration
```
function AGet(WPAT_Code: Integer; var Value: Integer): WordBool;
```

Description
Low level method to read a property referenced by a WPAT id.

8.15.2.3 IWPNumberStyle.ASet

Applies to
IWPNumberStyle 328

Declaration
```
procedure ASet(WPAT_Code: Integer; Value: Integer);
```

Description
Low level method to modify a property referenced by a WPAT id.

8.15.2.4 IWPNumberStyle.Command

Applies to
IWPNumberStyle 328

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.

8.15.2.5 IWPNumberStyle.DeleteStyle

Applies to
IWPNumberStyle 328

Declaration
```
procedure DeleteStyle;
```

Description
Delete this style. The interface must not be used afterwards.
We recommend to use GetNumberStyle(ID,-1000,level) 183 to delete styles.

8.15.2.6 IWPNumberStyle.SelectStyle

Applies to
IWPNumberStyle 328

Declaration
```
procedure SelectStyle;
```

Description
Make the style the current style referenced by CurrStyle 165.

8.16 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 166) the page size defined using PageSize 167 or in the OnMeasurePage is overridden.

The editor can be switched into the WordWrap 175 mode - then the text is formatted to exactly fit into the window horizontally.

Tip: To create a new section in the text you can use IWPTextCursor.InputSection 247.

```
IWPPageSize sect;
sect = wpdllInt1.TextCursor.InputSection(1);
sect.Landscape = true;
```

```
wpdllInt1.ReleaseInt(sect);
```

Note: The select flag (use SetProp 336 to set it and GetProp 336 to read it) will be automatically set when a property is changed.

Properties	Methods
BottomMargin 334	**GetProp** 336
Landscape 334	**SetPageWH** 337
LeftMargin 334	**SetProp** 336
MarginFooter 335	
MarginHeader 335	
MarginMirror 335	
PageHeight 335	
PageWidth 335	
RightMargin 336	
TopMargin 336	

PAR-MAXLENGTH-Tip: You can use the method Memo.TextCommand(4,N,0) 208 to specify that a maximum of N characters should be printed in one line.

8.16.1 Properties

8.16.1.1 BottomMargin

Applies to
IWPPageSize 333

Declaration
```
int BottomMargin;
```

Description
The margin area at the bottom of the page.

8.16.1.2 Landscape

Applies to
IWPPageSize 333

Declaration
```
bool Landscape;
```

Description
The orientaion of the page.

8.16.1.3 LeftMargin

Applies to
IWPPageSize 333

Declaration
```
int LeftMargin;
```

Description

The margin area at the left side of the page.

8.16.1.4 MarginFooter

Applies to
IWPPageSize 333

Declaration
```
int MarginFooter;
```

Description
This is the margin between the border of the page and the footer text.

8.16.1.5 MarginHeader

Applies to
IWPPageSize 333

Declaration
```
int MarginHeader;
```

Description
This is the margin between the border of the page and the header text.

8.16.1.6 MarginMirror

Applies to
IWPPageSize 333

Declaration
```
bool MarginMirror;
```

Description
If this property is true the left and right margin will be swapped on every otrher page.

8.16.1.7 PageHeight

Applies to
IWPPageSize 333

Declaration
```
int PageHeight;
```

Description
The page height measured in twips.

8.16.1.8 PageWidth

Applies to
IWPPageSize 333

Declaration
```
int PageWidth;
```

Description

The page width measured in twips.

8.16.1.9 RightMargin

Applies to
IWPPageSize 333

Declaration
```
int RightMargin;
```

Description
The margin area at the right side of the page.

8.16.1.10 TopMargin

Applies to
IWPPageSize 333

Declaration
```
int TopMargin;
```

Description
The margin area at the top of the page.

8.16.2 Methods

8.16.2.1 IWPPageSize.GetProp

Applies to
IWPPageSize 333

Declaration
```
function GetProp(Id: Integer): Integer;
```

Description

Use **ID=1** to retrieve a bitfield 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

8.16.2.2 IWPPageSize.SetProp

Applies to
IWPPageSize 333

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!

8.16.2.3 IWPPageSize.SetPageWH

Applies to
IWPPageSize 333

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.

8.16.2.4 IWPPageSizeList.Add

Applies to
IWPPageSizeList 338

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 339=1440).

8.16.2.5 IWPPageSizeList.AddEx

Applies to
IWPPageSizeList 338

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 339=1440).

8.16.2.6 IWPPageSizeList.Clear

Applies to

IWPPageSizeList 338

Declaration
```
procedure Clear;
```

Description
Clears all page sizes.

8.16.2.7 IWPPageSizeList.Delete

Applies to
IWPPageSizeList 338

Declaration
```
procedure Delete(Index: Integer);
```

Description
Delete a page size element with a given number. The first element has the index 0.

8.17 IWPPageSizeList

Description
This interface is used by PageSizeList 167.

You can use it to install page sizes for each page. To retrieve a page as metafile use GetPageAsMetafile 184.

Please see this demo 85.

8.17.1 Properties

8.17.1.1 AsString

Applies to
IWPPageSizeList 338

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.

8.17.1.2 Count

Applies to
IWPPageSizeList 338

Declaration
```
int Count;
```

Description
The count of defined page sizes - readonly.

8.17.1.3 LastPageHeight

Applies to
IWPPageSizeList 338

Declaration
```
int LastPageHeight;
```

Description
The text height of the last page - readonly.

8.17.1.4 LastPageMaxHeight

Applies to
IWPPageSizeList 338

Declaration
```
bool LastPageMaxHeight;
```

Description
If true (default) the last page uses a very large height and the width of the previous page. The way the last page will hold all text which did not fit into the previous pages.

8.17.1.5 Resolution

Applies to
IWPPageSizeList 338

Declaration
```
int Resolution;
```

Description
This is the resolution for the values provided to Add 337 and AddEx 337.

8.17.1.6 Active

Applies to
IWPPageSizeList 338

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.

8.18 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 219 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 301).

This codes appends text to the paragraph:

```
IWPAttrInterface 276 atr = wpdllInt1.AttrHelper;
IWPParInterface par = memo.CurrPar 164;
// Append normal text
atr.Clear();
par.AppendText 351("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 164;
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 219.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 196("New_Style");
// get interface to change the style
style = wpdllInt1.Memo.CurrStyle;
// set properties in stlye
style.Alignment 341 = 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 349:

```
// 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 339 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();
```

Properties:	Methods	Methods
Alignment 341	AppendChild 350	ParAAddBits 360
AlignmentVert 341	AppendNext 351	ParAClear 360
Borders 342	AppendText 351	ParADel 361
CellCommand 342	CharAttr 352	ParADelBits 361

8.18.1 Properties

8.18.1.1 Alignment

Applies to
IWPParInterface 339

Declaration
```
int Alignment;
```

Description
The alignment (WPAT_Alignment 413) used by this paragraph or style:
0 : Left aligned
1 : Centered
2 : Right aligned
3 : Justified Text

8.18.1.2 AlignmentVert

Applies to
IWPParInterface 339

Declaration

```
int AlignmentVert;
```

Description
Vertical alignment (WPAT_VertAlignment) - only possible in cells:
0 : top aligned
1 : centered vertically
2 : bottom aligned

8.18.1.3 Borders

Applies to
IWPParInterface 339

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.

8.18.1.4 CellCommand

Applies to
IWPParInterface 339

Declaration
```
string CellCommand;
```

Description
The formula used by this cell. Used for calculation in tables.

8.18.1.5 CellName

Applies to
IWPParInterface 339

Declaration
```
string CellName;
```

Description
The cell name used by this cell. Used for calculation in tables.

8.18.1.6 CharCount

Applies to
IWPParInterface 339

Declaration
```
int CharCount;
```

Description
The count of characters in a paragraph.

8.18.1.7 IndentFirst

Applies to
IWPParInterface 339

Declaration
```
int IndentFirst;
```

Description
The first indent (WPAT_IndentFirst) as twips value.

8.18.1.8 IndentLeft

Applies to
IWPParInterface 339

Declaration
```
int IndentLeft;
```

Description
The left indent (WPAT_IndentLeft) as twips value. Use IndentFirst=-IndentLeft for hanging indents.

8.18.1.9 IndentRight

Applies to
IWPParInterface 339

Declaration
```
int IndentRight;
```

Description
The right indent (WPAT_IndentRight) as twips value.

8.18.1.10 IsColMerge

Applies to
IWPParInterface 339

Declaration
```
bool IsColMerge;
```

Description
When set, this paragraph (must be a cell) is horizontally merged with the previous cell.

Please also see InputRowStart 246 - there we added an VB.NET example which creates a table using merged cells and paragraph styles.

8.18.1.11 IsFooterRow

Applies to
IWPParInterface 339

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 158

8.18.1.12 IsHeaderRow

Applies to
IWPParInterface 339

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 158

8.18.1.13 IsHidden

Applies to
IWPParInterface 339

Declaration
```
bool IsHidden;
```

Description
If set this paragraph (must not be a cell) is hidden.

8.18.1.14 IsNewPage

Applies to
IWPParInterface 339

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.

8.18.1.15 IsProtected

Applies to
IWPParInterface 339

Declaration
```
bool IsProtected;
```

Description

If set this paragraph is protected.

8.18.1.16 IsRowMerge

Applies to
IWPParInterface 339

Declaration
```
bool IsRowMerge;
```

Description
When set, this paragraph (must be a cell) is vertically merged with the previous cell.

Also see property IsColMerge 343.

Category
Table Support 158

8.18.1.17 LineHeight

Applies to
IWPParInterface 339

Declaration
```
int LineHeight;
```

Description
The line height in percent (WPAT_LineHeight) defined for this paragraph or style.

8.18.1.18 NumberLevel

Applies to
IWPParInterface 339

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;
```

8.18.1.19 NumberMode

Applies to
IWPParInterface 339

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 183.

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 183(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");
```

8.18.1.20 ParColor

Applies to
IWPParInterface 339

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 347 = 20
```

8.18.1.21 ParCSS

Applies to
IWPParInterface 339

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 159

8.18.1.22 ParShading

Applies to
IWPParInterface 339

Declaration
```
int ParShading;
```

Description
The paragraph shading as percent value. Also see property ParColor 347.

8.18.1.23 ParWPCSS

Applies to
IWPParInterface 339

Declaration
```
string ParWPCSS;
```

Description
Read and write the paragraph attributes in WPCSS format.

Category
TextDynamic CSS strings 159

8.18.1.24 SpaceAfter

Applies to
IWPParInterface 339

Declaration
```
int SpaceAfter;
```

Description
The space after the paragraph as twips value.

8.18.1.25 SpaceBefore

Applies to
IWPParInterface 339

Declaration
```
int SpaceBefore;
```

Description
The space before the paragraph as twips value.

8.18.1.26 SpaceBetween

Applies to
IWPParInterface 339

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.

8.18.1.27 StyleName

Applies to
IWPParInterface 339

Declaration
```
string StyleName;
```

Description
The name of the paragraph style sued be this paragraph. For paragraph styles this is the name of the base style.

You can add a style using Memo.SelectStyle 196(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;
}
```

Note: You need to call Memo.ReformatAll 191(true, true) after you have applied styles!

Category
Paragraphstyle Support 157

8.18.1.28 TOCOutlineLevel

Applies to
IWPParInterface 339

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)

8.18.1.29 WidthTW

Applies to
IWPParInterface 339

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 362.

8.18.2 Methods

8.18.2.1 Convert Utility function

The following methods can be used to create number values for the method ParASet 362 or to convert the result of ParAGet 361.

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 412.

8.18.2.2 IWPParInterface.AppendChild

Applies to
IWPParInterface 339

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 366 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 158

8.18.2.3 IWPParInterface.AppendNext

Applies to
IWPParInterface 339

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 350.

8.18.2.4 IWPParInterface.AppendText

Applies to
IWPParInterface 339

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 351("Some bold text", -2);
```

8.18.2.5 IWPParInterface.CharAttr

The IWPAttrInterface to modify one character attribute in a paragraph.

Applies to
IWPParInterface 339

Declaration
```
function CharAttr(Index: Integer): IWPAttrInterface 276;
```

Description
This method returns an IWPAttrInterface 276 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 365 to apply it.

Category
Character Attributes 146

8.18.2.6 IWPParInterface.CharObj

Applies to
IWPParInterface 339

Declaration
```
function CharObj(Index: Integer): IWPTextObj 396;
```

Description
This function provides a IWPTextObj 396 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.

8.18.2.7 IWPParInterface.ClearCharAttr

Applies to
IWPParInterface 339

Declaration
```
procedure ClearCharAttr;
```

Description
Sets the CharAttr ids of all the contained text and all children is set to to NEUTRAL. Also See ParAClear() 360.

8.18.2.8 IWPParInterface.DeleteChar

Applies to
IWPParInterface 339

Declaration
```
procedure DeleteChar(Index: Integer; Count: Integer);
```

Description
Delete N characters at a certain position.

8.18.2.9 IWPParInterface.DeleteParagraph

Applies to
IWPParInterface 339

Declaration
```
procedure DeleteParagraph;
```

Description
This method deletes this paragraph and all children. It makes the next paragraph the current.

8.18.2.10 IWPParInterface.DeleteParEnd

Applies to
IWPParInterface 339

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.

8.18.2.11 IWPParInterface.Duplicate

Applies to
IWPParInterface 339

Declaration
```
procedure Duplicate;
```

Description
This method duplicates this paragraph and all its text and children. It makes the new paragraph the current.

8.18.2.12 IWPParInterface.GetAllText

Applies to
IWPParInterface 339

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.

8.18.2.13 IWPParInterface.GetChar

Applies to
IWPParInterface 339

Declaration
```
function GetChar(Index: Integer): Word;
```

Description
This function returns the character at a certain position.

8.18.2.14 IWPParInterface.GetCharAttr

Applies to
IWPParInterface 339

Declaration
```
function GetCharAttr(Index: Integer): Integer;
```

Description
This method ready the attribute index at a certain position in the paragraph.

Category
Character Attributes 146

8.18.2.15 IWPParInterface.GetParType

Applies to
IWPParInterface 339

Declaration
```
function GetParType: Integer;
```

Description
This function returns the paragraph type. Regular paragraphs use the type 0, tables 6 and table rows 7.

8.18.2.16 IWPParInterface.GetProp

Applies to
IWPParInterface 339

Declaration
```
function GetProp(ID: Integer): Integer;
```

Description
This method is reserved to set additional paragraph properties.

8.18.2.17 IWPParInterface.GetPtr

Applies to
IWPParInterface 339

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 155

8.18.2.18 IWPParInterface.GetPtrChild

Applies to
IWPParInterface 339

Declaration
```
function GetPtrChild: Integer;
```

Description
This is the ID of the first child of this paragraph.

Category
Lowlevel Paragraph IDs 155

8.18.2.19 IWPParInterface.GetPtrNext

Applies to
IWPParInterface 339

Declaration
```
function GetPtrNext: Integer;
```

Description
This is the ID of the next sibling of the paragraph.

Category
Lowlevel Paragraph IDs 155

8.18.2.20 IWPParInterface.GetPtrParent

Applies to
IWPParInterface 339

Declaration
```
function GetPtrParent: Integer;
```

Description
This is the ID of the parent of the paragraph. You can use this value with SetPtr 366.

Category
Lowlevel Paragraph IDs 155

8.18.2.21 IWPParInterface.GetPtrPrev

Applies to
IWPParInterface 339

Declaration
```
function GetPtrPrev: Integer;
```

Description
This is the ID of the previous sibling of the paragraph.

Category
Lowlevel Paragraph IDs 155

8.18.2.22 IWPParInterface.GetSubText

Applies to
IWPParInterface 339

Declaration
```
function GetSubText(PosInPar: Integer; Len: Integer): WideString;
```

Description
You can use this method to extract a piece of the paragraph as a string.

8.18.2.23 IWPParInterface.GetText

Applies to
IWPParInterface 339

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,..) 362.

8.18.2.24 IWPParInterface.HasText

Applies to
IWPParInterface 339

Declaration
```
function HasText(const Text: WideString; CaseSensitive: WordBool): WordBool;
```

Description
This functions checks if a given text exists in this paragraph.

8.18.2.25 IWPParInterface.InsertNewObject

Applies to
IWPParInterface 339

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 127 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 149

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();
}
```

8.18.2.26 IWPParInterface.InsertText

Applies to
IWPParInterface 339

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)

8.18.2.27 IWPParInterface.LoadFromFile

Applies to
IWPParInterface 339

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 150

8.18.2.28 IWPParInterface.LoadFromString

Applies to
IWPParInterface 339

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 150

8.18.2.29 IWPParInterface.ParAAddBits

Applies to
IWPParInterface 339

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 353 to normalize a complete paragraph.

8.18.2.30 IWPParInterface.ParAClear

Applies to
IWPParInterface 339

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 353 this does not change children paragraphs!)

Category
Attribute IDs 410

8.18.2.31 IWPParInterface.ParADel

Applies to
IWPParInterface 339

Declaration
```
procedure ParADel(WPAT_Code: Integer);
```

Description
Deletes a certain attribute. This makes this attribute undefined.

Also see Convert 349 utility for Fontface & Color Values.

See: Attribute IDs 410

8.18.2.32 IWPParInterface.ParADelBits

Applies to
IWPParInterface 339

Declaration
```
procedure ParADelBits(WPAT_Code: Integer; Bits: Integer);
```

Description
Performs "AND NOT value" operation with specified value to delete bits.

Category
Attribute IDs 410

8.18.2.33 IWPParInterface.ParAGet

Applies to
IWPParInterface 339

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 349".

Category
Attribute IDs 410

8.18.2.34 IWPParInterface.ParAInc

Applies to
IWPParInterface 339

Declaration

```
procedure ParAInc(WPAT_Code: Integer; Offset: Integer; MinValue: Integer);
```

Description
Increments attribute. The last parameter is the minimum allowed value.

Category
Attribute IDs 410

8.18.2.35 IWPParInterface.ParASet

Applies to
IWPParInterface 339

Declaration
```
procedure ParASet(WPAT_Code: Integer; Value: Integer);
```

Description
Sets an attribute.

Also see Convert 349 Utility for Fontface & Color Values.

See: Attribute IDs 410

8.18.2.36 IWPParInterface.ParCommand

Applies to
IWPParInterface 339

Declaration
```
int ParCommand(int ComID, int param, int param2);
```

Description
Execute special commands for this paragraph, cell or table row.

ComID = 1:

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:

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:

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:

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 179)
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

8.18.2.37 IWPParInterface.ParStrCommand

Applies to
IWPParInterface 339

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 362. (Better use convert utility 349)

2: Convert a string into a number, use it with the code WPAT_NumberTEXTB and WPAT_NumberTEXTA. (Better use convert utility 349)

3: Convert a color string into a number, use it with the code WPAT_CharColor. (Better use convert utility 349)

4: Assigns the name of this paragraph

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 233 and SetParName 261.

Category
Attribute IDs 410

8.18.2.38 IWPParInterface.ReplaceCharAttr

Applies to
IWPParInterface 339

Declaration
```
procedure ReplaceCharAttr(PosInPar: Integer; Len: Integer; NewCharAttrIndex:
Integer);
```

Description
Replaces certain CharAttr index values with outhers.

Category
Character Attributes 146

8.18.2.39 IWPParInterface.ReplaceText

Applies to
IWPParInterface 339

Declaration
```
procedure ReplaceText(PosInPar: Integer; Len: Integer; const NewText:
WideString);
```

Description
Replaces a subtext with a different text.

8.18.2.40 IWPParInterface.SaveToString

Applies to
IWPParInterface 339

Declaration
```
function SaveToString(const Format: WideString; OnlyChildren: WordBool):
WideString;
```

Description
Saves the contents of zthe paragraph to a string. Also see LoadFromString 359.

Category
Load and Save 150

8.18.2.41 IWPParInterface.SetChar

Applies to
IWPParInterface 339

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.

8.18.2.42 IWPParInterface.SetCharAttr

Applies to
IWPParInterface 339

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 146

8.18.2.43 IWPParInterface.SetParType

Applies to
IWPParInterface 339

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

The .NET assembly defines the enum "ParagraphType".

Category
Table Support 158

8.18.2.44 IWPParInterface.SetProp

Applies to
IWPParInterface 339

Declaration

```
procedure SetProp(ID: Integer; Value: Integer);
```

Description
Reserved

8.18.2.45 IWPParInterface.SetPtr

Applies to
IWPParInterface 339

Declaration

```
bool SetPtr(int Paragraph);
```

Description
Makes the paragraph with the specified ID the 'current' paragraph. See method AppendChild 350 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 368" which make it easy to move through a document without changing the cursor position!

Category
Lowlevel Paragraph IDs 155

8.18.2.46 IWPParInterface.SetText

Applies to
IWPParInterface 339

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

8.18.2.47 IWPParInterface.TabAdd

Applies to
IWPParInterface 339

Declaration
```
procedure TabAdd(Twips: Integer; Kind: Integer; Fill: Integer; ColorNr:
Integer);
```

Description
Add (define) a tabstop in a paragraph.

To insert a paragraph at the current position use TextCursor.InputTabStop 250.

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 349 method)

8.18.2.48 IWPParInterface.TabClear

Applies to
IWPParInterface 339

Declaration
```
procedure TabClear;
```

Description
Clears all tabstops

8.18.2.49 IWPParInterface.TabDelete

Applies to
IWPParInterface 339

Declaration
```
void TabDelete(int Twips);
```

Description
Delete the tabstop at the specified position (twips value).

8.18.2.50 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
     Cell 3 line 2
     Cell 3 line 3

```
  </td>
 </tr>
</table>
```

Note:
a) Please also see SetPtr 366 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.

8.19 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);
```

8.20 IWPPrintParameter

Optimize printing (duplex, tray selection)

Description
Change Printing Options. You can use property PageList 371 to print a list of pages, or print the odd or even pages only (PageSides).

8.20.1 Properties

8.20.1.1 AllPagePaperSource

Applies to
IWPPrintParameter 370

Declaration
```
int AllPagePaperSource;
```

Description
The paper tray ID used for all pages.

8.20.1.2 DontUpdateDEVMode

Applies to
IWPPrintParameter 370

Declaration
```
bool DontUpdateDEVMode;
```

Description
If true TextDynamic will not try to modify printer settings.

8.20.1.3 DuplexMode

Applies to
IWPPrintParameter 370

Declaration
```
int DuplexMode;
```

Description
0=DontChangeDuplex
1=None
2=orizontal
3=Vertical

8.20.1.4 FirstPagePaperSource

Applies to
IWPPrintParameter 370

Declaration
```
int FirstPagePaperSource;
```

Description
The paper tray ID used for first page only.

8.20.1.5 Options

Applies to
IWPPrintParameter 370

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

8.20.1.6 PageList

Applies to
IWPPrintParameter 370

Declaration
```
string PageList;
```

Description
Page list as string, i.e "1-5,8,10"

8.20.1.7 PageSides

Applies to
IWPPrintParameter 370

Declaration
```
int PageSides;
```

Description
0=print all, 1=print even, 2=print odd pages only

8.20.1.8 Title

Applies to
IWPPrintParameter 370

Declaration
```
string Title;
```

Description

Title for the printer job.

8.20.2 Methods

8.20.2.1 IWPPrintParameter.SetExtraProp

Applies to
IWPPrintParameter 370

Declaration
```
procedure SetExtraProp(ID: Integer; Value: Integer);
```

8.21 IWPReport

Manage reporting template

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 pre-template. This pre template 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 pre-template 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". 87

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;`

This definition 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 pre-template 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 pre-template 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 pre-template 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

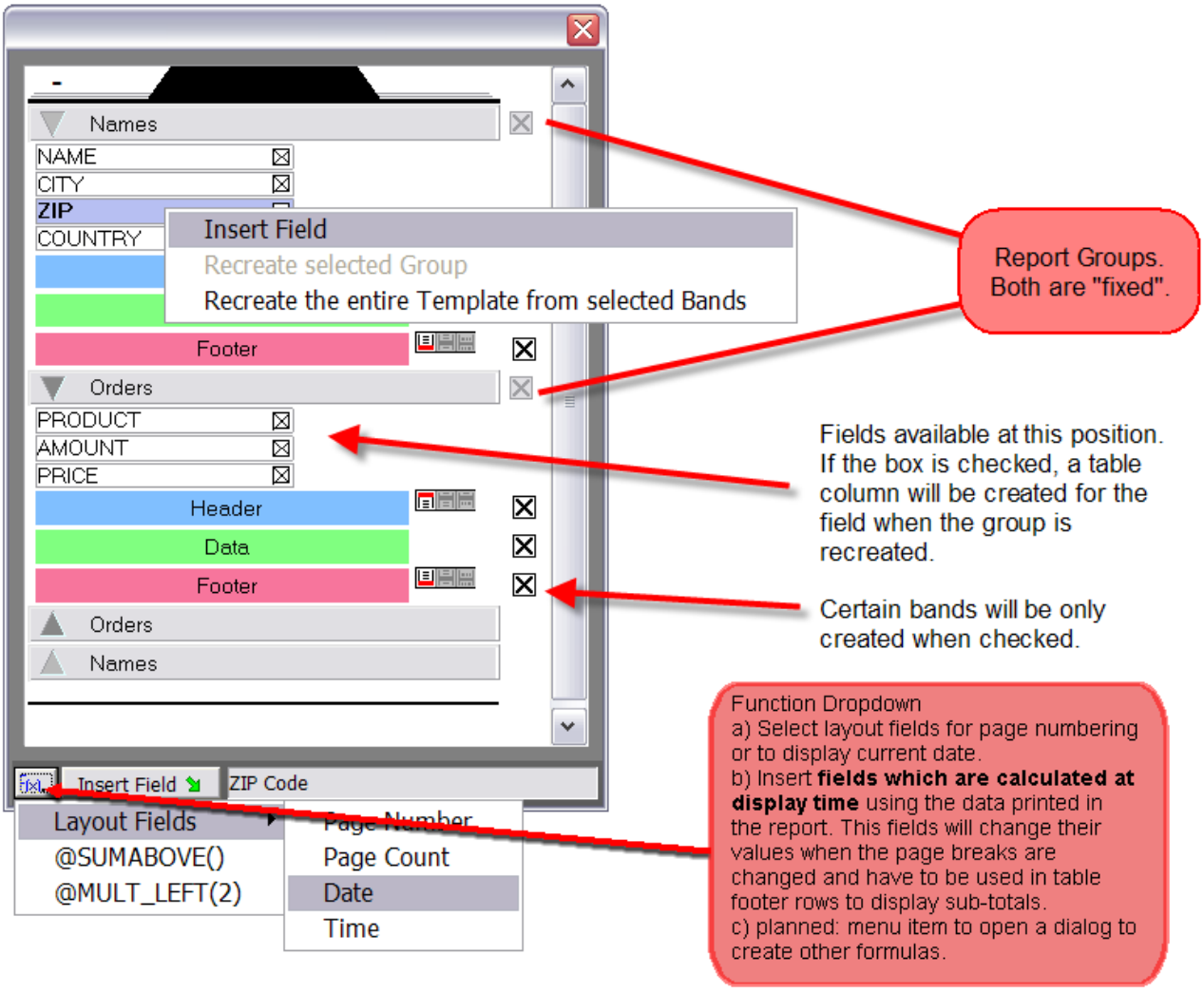
```
Report.InitTemplate("c:\\a_report_template.rtf",2+4);
```

Only set caption for pre-template 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	**AddSection**
RepositoryCaption	**AddText**
ShowRepository	**AddVar**

TemplateEditorStyle	**Band** **Clear** **Command** **CreateReport** DefinePageSize **FindGroup** **GetError** **GetErrorCount** InitGroup **InitTemplate** LoadFromFile **ModifyXML** **SaveToFile** **SelectSection** **SetAutomatic** **SetProp** **ShowResult** **ShowTemplate**

8.21.1 XML Template-Template

The interface "Reporter" manages a reporting template defined in XML. The XML structure is used to define and validate the variables.

This is the XML data which was created in the first example 89. 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 obligaotory 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.

8.21.2 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.

8.21.3 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.

8.21.4 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 375)

8.21.5 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.

8.21.6 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)".

8.22 IWPReportBand

Manage report bands and groups

Description
This interface is used during report creation inside the OnReportState event.

Properties

Methods

8.22.1 Properties

8.22.1.1 Alias

Applies to
IWPReportBand 376

Declaration
```
string Alias;
```

Description

This parameter can be used for an alternative alias name.

8.22.1.2 BandCount

Applies to
IWPReportBand 376

Declaration
```
int BandCount;
```

Description
Count of sub bands and groups. For sample code please see Report.Band(). The property is readonly.

8.22.1.3 Count

Applies to
IWPReportBand 376

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.

8.22.1.4 DBParams

Applies to
IWPReportBand 376

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.

8.22.1.5 Depth

Applies to
IWPReportBand 376

Declaration
```
int Depth;
```

Description
The nesting level of this band. The property is readonly.

8.22.1.6 DisplayName

Applies to
IWPReportBand 376

Declaration
```
string DisplayName;
```

Description
This name should be displayed in the editor.

8.22.1.7 GroupRadioNr

Applies to
IWPReportBand 376

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 378 as this band.

8.22.1.8 GroupSiblingNr

Applies to
IWPReportBand 376

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.

8.22.1.9 Mode

Applies to
IWPReportBand 376

Declaration
```
int Mode;
```

Description
Reserved Property.

8.22.1.10 Name

Applies to
IWPReportBand 376

Declaration
```
string Name;
```

Description
The name of this group or band. Usually only group have names.

8.22.1.11 Options

Applies to

IWPReportBand 376

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.

8.22.1.12 ParAttr

Applies to
IWPReportBand 376

Declaration
IWPAttrInterface 276 ParAttr;

Description
Currently not used.

8.22.1.13 ParentGroupCount

Applies to
IWPReportBand 376

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.

8.22.1.14 ParentGroupName

Applies to
IWPReportBand 376

Declaration
```
string ParentGroupName;
```

Description
This is the name of the parent group.

8.22.1.15 ParentParentGroup

Applies to
IWPReportBand 376

Declaration
```
string ParentParentGroup;
```

Description

This is the name of the parent group of the parent group.

8.22.1.16 ParentParentGroupCount

Applies to
IWPReportBand 376

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.

8.22.1.17 RecordSet

Applies to
IWPReportBand 376

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.

8.22.1.18 Selected

Applies to
IWPReportBand 376

Declaration
```
bool Selected;
```

Description
This property is used to change the selected state. If the Visibility 381 bit 5 (=16) is set, a band will only be used if selected. Using the properties GroupRadioNr 378 and GroupSiblingNr 378 bands on different levels can be combined to logical groups.

8.22.1.19 State

Applies to
IWPReportBand 376

Declaration
```
int State;
```

8.22.1.20 Typ

Applies to
IWPReportBand 376

Declaration
```
int Typ;
```

8.22.1.21 Visibility

Applies to
IWPReportBand 376

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.

8.22.2 Methods

8.22.2.1 IWPReportBand.AddBand

Applies to
IWPReportBand 376

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.

8.22.2.2 IWPReportBand.AddTable

Applies to
IWPReportBand 376

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 158

8.22.2.3 IWPReportBand.Band

Applies to
IWPReportBand 376

Declaration
```
function Band(Index: Integer): IWPReportBand;
```

Description
Retrieve sub bands. For sample code please see Report.Band().

8.22.2.4 IWPReportBand.CheckSyntax

Applies to
IWPReportBand 376

Declaration
```
function CheckSyntax: Integer;
```

Description
This method is not used.

8.22.2.5 IWPReportBand.Clear

Applies to
IWPReportBand 376

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.

8.22.2.6 IWPReportBand.FindVar

Applies to
IWPReportBand 376

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 384.

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 '-'.

8.22.2.7 IWPReportBand.GetParPtr

Applies to
IWPReportBand 376

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.

8.22.2.8 IWPReportBand.GetProp

Applies to
IWPReportBand 376

Declaration
```
function GetProp(Item: Integer): WideString;
```

Description
Reserved.

8.22.2.9 IWPReportBand.GetVar

Applies to
IWPReportBand 376

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 384.

8.22.2.10 IWPReportBand.ParentGroup

Applies to
IWPReportBand 376

Declaration
```
function ParentGroup: IWPReportBand;
```

Description
This functions returns either null or a reference of the parent group of this band.

8.22.2.11 IWPReportBand.SetParPtr

Applies to
IWPReportBand 376

Declaration
```
procedure SetParPtr(Paragraph: Integer);
```

Description
This method may not be used.

8.22.2.12 IWPReportBand.SetProp

Applies to
IWPReportBand 376

Declaration
```
procedure SetProp(Item: Integer; const Value: WideString);
```

Description
Reserved

8.22.2.13 IWPReportBand.VarCount

Applies to
IWPReportBand 376

Declaration
```
function VarCount: Integer;
```

Description
This function returns the count of variables of a group.

8.23 IWPReportVar

Manage reporting variables (to sum values)

Description
This technology is under development. It is expected to be ready by Mid July 2006.

Properties

- **Description** 385
- **Format** 385
- GetTextFormula 385
- **LoopFormula** 385
- Mode 385
- Name 386
- ParStyle 386
- **StartFormula** 386
- **Text** 386
- **Title** 386
- **Value** 387

Methods

- **ElementsAdd** 387
- **ElementsClear** 387
- **ElementsCount** 387
- **ElementsGet** 388

WidthTW 387

8.23.1 Properties

8.23.1.1 Description

Applies to
IWPReportVar 384

Declaration
```
string Description;
```

Description
This is the description for this variable.

8.23.1.2 Format

Applies to
IWPReportVar 384

Declaration
```
string Format;
```

Description
This is the format string. "CUR=$" will format as currency.

8.23.1.3 GetTextFormula

Applies to
IWPReportVar 384

Declaration
```
string GetTextFormula;
```

8.23.1.4 LoopFormula

Applies to
IWPReportVar 384

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.

8.23.1.5 Mode

Applies to
IWPReportVar 384

Declaration
```
int Mode;
```

8.23.1.6 Name

Applies to
IWPReportVar 384

Declaration
```
string Name;
```

8.23.1.7 ParStyle

Applies to
IWPReportVar 384

Declaration
```
string ParStyle;
```

8.23.1.8 StartFormula

Applies to
IWPReportVar 384

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 385.

8.23.1.9 Text

Applies to
IWPReportVar 384

Declaration
```
string Text;
```

Description
This is the value as text.

8.23.1.10 Title

Applies to
IWPReportVar 384

Declaration
```
string Title;
```

Description
The title of this variable.

8.23.1.11 Value

Applies to
IWPReportVar 384

Declaration
```
double Value;
```

Description
The value of this variable.

8.23.1.12 WidthTW

Applies to
IWPReportVar 384

Declaration
```
int WidthTW;
```

Description
Reserved.

8.23.2 Methods

8.23.2.1 IWPReportVar.ElementsAdd

Applies to
IWPReportVar 384

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 387. 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.

8.23.2.2 IWPReportVar.ElementsClear

Applies to
IWPReportVar 384

Declaration
```
procedure ElementsClear;
```

Description
Removes all elements and sets the value=0.

8.23.2.3 IWPReportVar.ElementsCount

Applies to
IWPReportVar 384

Declaration
```
function ElementsCount: Integer;
```

Description
This is the number of elements.

8.23.2.4 IWPReportVar.ElementsGet

Applies to
IWPReportVar 384

Declaration
```
function ElementsGet(Index: Integer; var Value: Double; var Color: Integer):
WideString;
```

Description
Retrieve a certain element.

8.24 IWPSpell

Interface to modify properties of the optional spell check feature.

Description
You can use this interface to load dictionaries from certain directories, select the language which is currently used and load or save the current setup to registry or INI file.

If you do not use this interface the setup will be load and saved to the registry key Software \WPCubed\WPSpell.
You can use the method SetSetupPersistency to manually select registry and time when the setup is loaded.

Start - Stop Spellcheck:

Use the method TextCommand(7,mode) 208 to start and stop spllcheck.

The following values are allowed for mode:
0=Start SpellCheck (using the internal spellchecker),
1= Start Thesuarus (reserved),
2=Start SpellAsYouGo (internal or external),
3=Stop SpellAsYouGo, (internal or external),
4=Show SpellCheckSetup (internal only).

This "wpa" actions can also be used, for example with wpaProcess 122(wpa_name):

```
"DiaSpellcheck",
"DiaSpellOptions",
"Spellcheck",
"SpellAsYouGo".
```

"SpellAsYouGo" can be called with parameter "1" to switch the online spell check (curly underlines) ON or with parameter "0" to switch it OFF. If no parameter is used the state will be toggled.

Example - VB6:

```
Private Sub Form_Load()
    On Error GoTo Error_Trap
```

```
    ' set license
    WPDLLInt1.EditorStart 110 "your license name", "your license code"

    ' load PCC file
    WPDLLInt1.SetLayout 116 App.Path & "\Buttons.PCC", "Default", "", "main", "main"

    ' initialize editor
    WPDLLInt1.SetEditorMode 114 0, 1 + 4 + 8, 2 + 4 + 8 + 16 + 64 + 128 + 256 + 1024, 0

    'Load Dictionary
    WPDLLInt1.SpellCtrl.AddFromFile 389 App.Path & "\English.DCT"

    ' Set "English" as standard Language
    WPDLLInt1.SpellCtrl.SetLanguage 392 "9" ' Select English

    ' Switch ON "Spell-As-You-Go"
    WPDLLInt1.wpaProcess 122 "SpellAsYouGo", "1"

    Form_Resize

Exit_Routine:
    On Error Resume Next
    Exit Sub
Error_Trap:
    ErrHandler Err.Number, Err.Description, EGO, "Form_Load"
    Resume Exit_Routine
End Sub
```

Tip: You can also use the method CommandEx 107(9500, n, strparam) to execute some of the methods. This may be useful if you cannot access the interface in your developing language.

Custom Spellchecking: You can also use your own check routine with TextDynamic, you will need **your own** dictionary API. This will disable the integrated spell check engine.

To activate custom spellchecking use Command(907) 107
Use a parameter=0 to disable custom spellcheck, use a parameter = A (any number) to enable it. The event OnEnumParOrStyle 127 will be triggered to check words with EventParam=A. Also see OnMouseDownWord 139 which can be used to create a popup menu. You should call Command(907) before SetEditorMode.

8.24.1 Methods

8.24.1.1 IWPSpell.AddFromFile

Applies to
IWPSpell 388

Declaration
```
bool AddFromFile(string filename)
```

Description
This method loads a dictionary "DCT file" from the give location. The function returns TRUE if the file was loaded alright. You can use the token {dll} to specify the file path relatively to the location of the TextDynamic word processing engine DLL.

Example VB6:

```
'Load Dictionary
WPDLLInt1.SpellCtrl.AddFromFile App.Path & "\english.dct"
```

8.24.1.2 IWPSpell.AddFromPath

Applies to
IWPSpell 388

Declaration
```
AddFromPath(string PathName)
```

Description
This method will add all *.DCT files from a given directory to the dictionary repository. You can use the token {dll} to specify the file path relatively to the location of the TextDynamic word processing engine DLL.

8.24.1.3 IWPSpell.AddWord

Applies to
IWPSpell 388

Declaration
```
procedure AddWord(const AWord: WideString; const aReplace: WideString; Mode:
Integer);
```

Description
This method adds a word to the user dictionary. The first parameter is the word, the second an optional replace text. The third parameter controls how the word can be used:
0 : this word is now known (added)
1 : this word is not known (excluded)
2 : auto replace
3 : delete the word from the list

8.24.1.4 IWPSpell.ClearAll

Applies to
IWPSpell 388

Declaration
```
procedure ClearAll;
```

Description
Removes all known dictionaries. Use AddFromFile or AddFromPath to add new dictionaries.

8.24.1.5 IWPSpell.Execute

Applies to
IWPSpell 388

Declaration
```
procedure Execute(Editor: Integer; Mode: Integer);
```

Description
Execute spellcheck for editor 1 or 2. In contrast to the toolbutton this will also work if spellcheck has not been enabled using the xmodes in the method SetEditorMode - the spell check licenses is required though.
The following modes are possible:
0 : Open spellcheck dialog
1 : Reserved

2 : Switch on Spell-As-You-Go (curly underlines)
3 : Disable the curly underlines
4 : Open setup dialog

8.24.1.6 IWPSpell.GetLanguage

Applies to
IWPSpell 388

Declaration
```
function GetLanguage: WideString;
```

Description
This method can be used to read the name of the language which is currently selected.

8.24.1.7 IWPSpell.GetLanguageName

Applies to
IWPSpell 388

Declaration
```
function GetLanguageName(Index: Integer): WideString;
```

Description
This method reads the language name of the currently selected dictionary.

8.24.1.8 IWPSpell.InDictionary

Applies to
IWPSpell 388

Declaration
```
function InDictionary(const Word: WideString): WordBool;
```

Description
This method checks if a certain word can be found.

8.24.1.9 IWPSpell.LoadSetup

Applies to
IWPSpell 388

Declaration
```
procedure LoadSetup;
```

Description
This method can be used to load the setup from either INI fiole or registry. (See SetSetupPersistency 395)

8.24.1.10 IWPSpell.SaveSetup

Applies to
IWPSpell 388

Declaration
```
procedure SaveSetup;
```

Description
This method can be used to save the setup to either INI fiole or registry. (See SetSetupPersistency 395)

8.24.1.11 IWPSpell.SetLanguage

Applies to
IWPSpell 388

Declaration
```
function SetLanguage(const NameOrNr: WideString): WordBool;
```

Description
This method can be used to select a different language. You can either use a name or a language id or group-id. If the dictionary was not found in the list of loaded dictionaries the result value will be false.

As language IDs the standard numbers defined by the RTF specifications are used. When you use the dictionary compiler (provided with "Spell" license) please set the correct language id. All language names have a special identifier and a group, for example all "English" flavors share the number 9. It is possible to select a language using the special ID or the group ID.

Example VB6:

```
' Set English as standard Language
WPDLLInt1.SpellCtrl.SetLanguage "9"
```

Widely used group IDs:

```
English:     9
French:      1036
German:      1031
Spanish:     1034
Italian:     1040
Dutch:       1043
```

Language IDs:

```
( (name:'Custom';id:0;groupid:0),
(name:'Afrikaans';id:1078;groupid:1078),
(name:'Albanian';id:1052;groupid:1052),
(name:'Arabic';id:1025;groupid:1025),
(name:'Arabic-Algeria';id:5121;groupid:1025),
(name:'Arabic-Bahrain';id:15361;groupid:1025),
(name:'Arabic-Egypt';id:3073;groupid:1025),
(name:'Arabic-General';id:1;groupid:1025),
(name:'Arabic-Iraq';id:2049;groupid:1025),
(name:'Arabic-Jordan';id:11265;groupid:1025),
(name:'Arabic-Kuwait';id:13313;groupid:1025),
(name:'Arabic-Lebanon';id:12289;groupid:1025),
(name:'Arabic-Libya';id:4097;groupid:1025),
(name:'Arabic-Morocco';id:6145;groupid:1025),
(name:'Arabic-Oman';id:8193;groupid:1025),
(name:'Arabic-Qatar';id:16385;groupid:1025),
(name:'Arabic-Syria';id:10241;groupid:1025),
(name:'Arabic-Tunisia';id:7169;groupid:1025),
(name:'Arabic-U.A.E.';id:14337;groupid:1025),
(name:'Arabic-Yemen';id:9217;groupid:1025),
(name:'Armenian';id:1067;groupid:1067),
(name:'Assamese';id:1101;groupid:1101),
(name:'Azeri-Cyrillic';id:2092;groupid:2092),
```

```
(name:'Azeri-Latin';id:1068;groupid:1068),
(name:'Basque';id:1069;groupid:1069),
(name:'Bengali';id:1093;groupid:1093),
(name:'Bosnia-Herzegovina';id:4122;groupid:4122),
(name:'Bulgarian';id:1026;groupid:1026),
(name:'Burmese';id:1109;groupid:1109),
(name:'Belorussian';id:1059;groupid:1059),
(name:'Catalan';id:1027;groupid:1027),
(name:'Chinese-China';id:2052;groupid:4),
(name:'Chinese-General';id:4;groupid:4),
(name:'Chinese-HongKong';id:3076;groupid:4),
(name:'Chinese-Macao';id:3076;groupid:4),
(name:'Chinese-Singapore';id:4100;groupid:4),
(name:'Chinese-Taiwan';id:1028;groupid:4),
(name:'Croatian';id:1050;groupid:1050),
(name:'Czech';id:1029;groupid:1029),
(name:'Danish';id:1030;groupid:1030),
(name:'Dutch-Belgium';id:2067;groupid:1043),
(name:'Dutch-Standard';id:1043;groupid:1043),
(name:'English-Australia';id:3081;groupid:9),
(name:'English-Belize';id:10249;groupid:9),
(name:'English-British';id:2057;groupid:9),
(name:'English-Canada';id:4105;groupid:9),
(name:'English-Caribbean';id:9225;groupid:9),
(name:'English-General';id:9;groupid:9),
(name:'English-Ireland';id:6153;groupid:9),
(name:'English-Jamaica';id:8201;groupid:9),
(name:'English-NewZealand';id:5129;groupid:9),
(name:'English-Philippines';id:13321;groupid:9),
(name:'English-SouthAfrica';id:7177;groupid:9),
(name:'English-Trinidad';id:11273;groupid:9),
(name:'English-US';id:1033;groupid:9),
(name:'English-Zimbabwe';id:1033;groupid:9),
(name:'Estonian';id:1061;groupid:1061),
(name:'Faeroese';id:1080;groupid:1080),
(name:'Farsi';id:1065;groupid:1065),
(name:'Finnish';id:1035;groupid:1035),
(name:'French';id:1036;groupid:1036),
(name:'French-Belgium';id:2060;groupid:1036),
(name:'French-Cameroon';id:11276;groupid:1036),
(name:'French-Canada';id:3084;groupid:1036),
(name:'French-CotedIvoire';id:12300;groupid:1036),
(name:'French-Luxemburg';id:5132;groupid:1036),
(name:'French-Mali';id:13324;groupid:1036),
(name:'French-Monaco';id:6156;groupid:1036),
(name:'French-Reunion';id:8204;groupid:1036),
(name:'French-Senegal';id:10252;groupid:1036),
(name:'French-Swiss';id:4108;groupid:1036),
(name:'French-WestIndies';id:7180;groupid:1036),
(name:'French-Zaire';id:9228;groupid:1036),
(name:'Frisian';id:1122;groupid:1122),
(name:'Gaelic';id:1084;groupid:1084),
(name:'Gaelic-Ireland';id:2108;groupid:1084),
(name:'Galician';id:1110;groupid:1110),
(name:'Georgian';id:1079;groupid:1079),
(name:'German';id:1031;groupid:1031),
(name:'German-Austrian';id:3079;groupid:1031),
(name:'German-Liechtenstein';id:5127;groupid:1031),
(name:'German-Luxemburg';id:4103;groupid:1031),
(name:'German-Switzerland';id:2055;groupid:1031),
(name:'Greek';id:1032;groupid:1032),
(name:'Gujarati';id:1095;groupid:1095),
(name:'Hebrew';id:1037;groupid:1037),
(name:'Hindi';id:1081;groupid:1081),
(name:'Hungarian';id:1038;groupid:1038),
(name:'Icelandic';id:1039;groupid:1039),
(name:'Indonesian';id:1057;groupid:1057),
(name:'Italian';id:1040;groupid:1040),
(name:'Italian-Switzerland';id:2064;groupid:1040),
(name:'Japanese';id:1041;groupid:1041),
(name:'Kannada';id:1099;groupid:1099),
(name:'Kashmiri';id:1120;groupid:1120),
(name:'Kashmiri-India';id:2144;groupid:1120),
(name:'Kazakh';id:1087;groupid:1087),
(name:'Khmer';id:1107;groupid:1107),
(name:'Kirghiz';id:1088;groupid:1088),
(name:'Konkani';id:1111;groupid:1111),
(name:'Korean';id:1042;groupid:1042),
(name:'Korean-Johab';id:2066;groupid:1042),
```

```
(name:'Lao';id:1108;groupid:1108),
(name:'Latvian';id:1062;groupid:1062),
(name:'Lithuanian';id:1063;groupid:1063),
(name:'Lithuanian-Classic';id:2087;groupid:1063),
(name:'Macedonian';id:1086;groupid:1086),
(name:'Malay';id:1086;groupid:1086),
(name:'Malay-BruneiDarussalam';id:2110;groupid:1086),
(name:'Malayalam';id:1100;groupid:1086),
(name:'Maltese';id:1082;groupid:1082),
(name:'Manipuri';id:1112;groupid:1112),
(name:'Marathi';id:1102;groupid:1102),
(name:'Mongolian';id:1104;groupid:1104),
(name:'Nepali';id:1121;groupid:1121),
(name:'Nepali-India';id:2145;groupid:1121),
(name:'Norwegian-Bokmal';id:1044;groupid:1044),
(name:'Norwegian-Nynorsk';id:2068;groupid:1044),
(name:'Oriya';id:1096;groupid:1096),
(name:'Polish';id:1045;groupid:1045),
(name:'Portuguese-Brazil';id:1046;groupid:2070),
(name:'Portuguese-Iberian';id:2070;groupid:2070),
(name:'Punjabi';id:1094;groupid:1094),
(name:'Rhaeto-Romanic';id:1047;groupid:1047),
(name:'Romanian';id:1048;groupid:1048),
(name:'Romanian-Moldova';id:2072;groupid:1048),
(name:'Russian';id:1049;groupid:1049),
(name:'Russian-Moldova';id:2073;groupid:1049),
(name:'Sami-Lappish';id:1083;groupid:1083),
(name:'Sanskrit';id:1103;groupid:1103),
(name:'Serbian-Cyrillic';id:3098;groupid:3098),
(name:'Serbian-Latin';id:2074;groupid:2074),
(name:'Sindhi';id:1113;groupid:1113),
(name:'Slovak';id:1051;groupid:1051),
(name:'Slovenian';id:1060;groupid:1060),
(name:'Sorbian';id:1070;groupid:1070),
(name:'Spanish-Argentina';id:11274;groupid:1034),
(name:'Spanish-Bolivia';id:16394;groupid:1034),
(name:'Spanish-Chile';id:13322;groupid:1034),
(name:'Spanish-Colombia';id:9226;groupid:1034),
(name:'Spanish-CostaRica';id:5130;groupid:1034),
(name:'Spanish-DominicanRepublic';id:7178;groupid:1034),
(name:'Spanish-Ecuador';id:12298;groupid:1034),
(name:'Spanish-ElSalvador';id:17418;groupid:1034),
(name:'Spanish-Guatemala';id:4106;groupid:1034),
(name:'Spanish-Honduras';id:18442;groupid:1034),
(name:'Spanish-Mexico';id:2058;groupid:1034),
(name:'Spanish-Modern';id:3082;groupid:1034),
(name:'Spanish-Nicaragua';id:19466;groupid:1034),
(name:'Spanish-Panama';id:6154;groupid:1034),
(name:'Spanish-Paraguay';id:15370;groupid:1034),
(name:'Spanish-Peru';id:10250;groupid:1034),
(name:'Spanish-PuertoRico';id:20490;groupid:1034),
(name:'Spanish';id:1034;groupid:1034), // Traditional
(name:'Spanish-Uruguay';id:14346;groupid:1034),
(name:'Spanish-Venezuela';id:8202;groupid:1034),
(name:'Sutu';id:1072;groupid:1072),
(name:'Swahili';id:1089;groupid:1089),
(name:'Swedish';id:1053;groupid:1053),
(name:'Swedish-Finland';id:2077;groupid:1053),
(name:'Tajik';id:1064;groupid:1064),
(name:'Tamil';id:1097;groupid:1097),
(name:'Tatar';id:1092;groupid:1092),
(name:'Telugu';id:1098;groupid:1098),
(name:'Thai';id:1054;groupid:1054),
(name:'Tibetan';id:1105;groupid:1105),
(name:'Tsonga';id:1073;groupid:1073),
(name:'Tswana';id:1074;groupid:1074),
(name:'Turkish';id:1055;groupid:1055),
(name:'Turkmen';id:1090;groupid:1090),
(name:'Ukrainian';id:1058;groupid:1058),
(name:'Urdu';id:1056;groupid:1056),
(name:'Urdu-India';id:2080;groupid:1056),
(name:'Uzbek-Cyrillic';id:2115;groupid:2115),
(name:'Uzbek-Latin';id:1091;groupid:1091),
(name:'Venda';id:1075;groupid:1075),
(name:'Vietnamese';id:1066;groupid:1066),
(name:'Welsh';id:1106;groupid:1106),
(name:'Xhosa';id:1076;groupid:1076),
(name:'Yiddish';id:1085;groupid:1085),
(name:'Zulu';id:1077;groupid:1077) );
```

8.24.1.12 IWPSpell.SetProperty

Applies to
IWPSpell 388

Declaration
```
function SetProperty(ID: Integer; Value: Integer): Integer;
```

Description
This method is reserved.

8.24.1.13 IWPSpell.SetSetupPersistency

Applies to
IWPSpell 388

Declaration
```
procedure SetSetupPersistency(Mode: Integer; const Path: WideString);
```

Description
You can use this method to change the way the spellcheck controller loads and saves its setup.

Parameters	
Mode	Select the mode how the setup is saved. Bit 1: If set an INI file is used, otherwise the registry. Bit 2: If clear the setup is automatically loaded and save - if set you have to call LoadSetup and SaveSetup.
Path	This is the registry path or the path of the INI file. When using INI files you can use the token {dll}.

8.24.1.14 IWPSpell.UserDictAdd

Applies to
IWPSpell 388

Declaration
```
procedure UserDictAdd(const filename: WideString);
```

Description
This method add a user dictionary.

8.24.1.15 IWPSpell.UserDictRemove

Applies to
IWPSpell 388

Declaration
```
procedure UserDictRemove(const Path: WideString);
```

Description
The method removes a user dictionary.

8.25 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();
```

8.26 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 165 or to check objects inside an event (OnEnumTxtObj 127) it can be very useful.

The property EmbeddedText 397 allows it to read and modify the text within paired objects, such as hyperlinks, bookmarks and mail merge fields.

Properties

- **Command** 397
- **Contents_Filename** 397
- **EmbeddedText** 397
- **Frame** 397
- **Height** 398
- **IntParam** 398
- **Mode** 398
- **Name** 398
- **ObjType** 399
- **Params** 399

Methods

- **_SetObjType** 401
- **Clear** 402
- **Contents_Edit** 402
- **Contents_Height** 402
- **Contents_SaveToFile** 403
- **Contents_Width** 403
- DeleteObj 403

Methods

- **GetPtr** 404
- **LoadFromFile** 405
- **LoadFromStream** 405
- **MakeEndTag** 406
- **MoveCursor** 406
- **MoveToPage** 406
- **ScaleSize** 407
- **Select** 407
- **SetContentsID** 407
- **SetEmbText** 408
- SetFieldProp 408

8.26.1 Properties

8.26.1.1 Command

Applies to
IWPTextObj 396

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.

8.26.1.2 Contents_Filename

Applies to
IWPTextObj 396

Declaration
```
string Contents_Filename;
```

Description
This is the file name of the image object linked to this object.

8.26.1.3 EmbeddedText

Applies to
IWPTextObj 396

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!

8.26.1.4 Frame

Applies to
IWPTextObj 396

Declaration
```
int Frame;
```

Description
If the object is an image you can specify a frame:
0 : no frame
1 : single frame
2 : think frame

8.26.1.5 Height

Applies to
IWPTextObj 396

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.

8.26.1.6 IntParam

Applies to
IWPTextObj 396

Declaration
```
int IntParam;
```

Description
Integer property can be used for custom parameters.

8.26.1.7 Mode

Applies to
IWPTextObj 396

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.

8.26.1.8 Name

Applies to
IWPTextObj 396

Declaration
```
string Name;
```

Description
This is the name of the objects, for example the fieldname of a mailmerge field.

8.26.1.9 ObjType

Applies to
IWPTextObj 396

Declaration
```
TextObjTypes ObjType;
```

Description

The type of the object. The property is readonly, You can use _SetObjType 401 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 413.

8.26.1.10 Params

Applies to
IWPTextObj 396

Declaration
```
string Params;
```

Description
This is the default text displayed by a textobject field.

8.26.1.11 PositionMode

Applies to
IWPTextObj 396

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 400 and RelY 400 are used to move the image using twips offset values.

8.26.1.12 RelX

Applies to
IWPTextObj 396

Declaration
```
int RelX;
```

Description
Used to position a movable image (PositionMode =1 or 2). It is the horizontal offset in twips.

8.26.1.13 RelY

Applies to
IWPTextObj 396

Declaration
```
int RelY;
```

Description
Used to position a movable image (PositionMode =1 or 2). It is the vertical offset in twips.

8.26.1.14 StyleName

Applies to
IWPTextObj 396

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 157

8.26.1.15 Width

Applies to
IWPTextObj 396

Declaration
```
int Width;
```

Description
This is the width of the object in twips (1/1440 inch). It is only used by images.

8.26.1.16 wpcss

Applies to
IWPTextObj 396

Declaration
```
string wpcss;
```

Description
Read and write the properties as WPCSS string.

8.26.1.17 Format

Applies to
IWPTextWriter 409

Declaration
```
string Format;
```

Description
The format options used when saving the file

8.26.1.18 Wrap

Applies to
IWPTextObj 396

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

8.26.2 Methods

8.26.2.1 IWPTextObj._SetObjType

Applies to
IWPTextObj 396

Declaration
```
procedure _SetObjType(Value: Integer);
```

Description
This method sets the type of the object. Use with care.

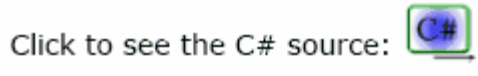
The following types are possible:
0=text object, 1=merge field, 2=hyperlink, 3=bookmark, 7=text object, 8=page reference, 11=footnote, 12=image or text box and 13 for horizontal lines.

> **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 236.
> 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");
```



8.26.2.2 IWPTextObj.Clear

Applies to
IWPTextObj 396

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).

8.26.2.3 IWPTextObj.Contents_Edit

Applies to
IWPTextObj 396

Declaration
```
function Contents_Edit: WordBool;
```

Description
This method selects the edit mode for footnotes and text boxes.

8.26.2.4 IWPTextObj.Contents_Height

Applies to
IWPTextObj 396

Declaration
```
function Contents_Height: Integer;
```

Description
This is the physical height of the loaded image.

8.26.2.5 IWPTextObj.Contents_LoadFromFile

Applies to
IWPTextObj 396

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.

8.26.2.6 IWPTextObj.Contents_SaveToFile

Applies to
IWPTextObj 396

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.

8.26.2.7 IWPTextObj.Contents_Width

Applies to
IWPTextObj 396

Declaration
```
function Contents_Width: Integer;
```

Description
This is the physical width of the loaded image.

8.26.2.8 IWPTextObj.DeleteObj

Applies to
IWPTextObj 396

Declaration
```
procedure DeleteObj;
```

8.26.2.9 IWPTextObj.GetContentsID

Applies to
IWPTextObj 396

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 407 to assign the object also to another placeholder which clones the image.

8.26.2.10 IWPTextObj.GetEmbText

Applies to
IWPTextObj 396

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.

8.26.2.11 IWPTextObj.GetFieldProp

Applies to
IWPTextObj 396

Declaration
```
function GetFieldProp(ID: Integer; var Value: WideString): WordBool;
```

8.26.2.12 IWPTextObj.GetParentParPos

Applies to
IWPTextObj 396

Declaration
```
function GetParentParPos: Integer;
```

Description
This is the position of the object inside the parent paragraph.

8.26.2.13 IWPTextObj.GetParentParPtr

Applies to
IWPTextObj 396

Declaration
```
function GetParentParPtr: Integer;
```

Description
This function retrieves a low level reference to the paragraph this object is located within.

8.26.2.14 IWPTextObj.GetProp

Applies to
IWPTextObj 396

Declaration
```
function GetProp(ID: Integer): WideString;
```

Description
This method is used to set the properties set by SetProp 408.

8.26.2.15 IWPTextObj.GetPtr

Applies to
IWPTextObj 396

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 406.

8.26.2.16 IWPTextObj.LoadFromFile

Applies to
IWPTextObj 396

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 236) 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 150
Image Support 149

8.26.2.17 IWPTextObj.LoadFromStream

Applies to
IWPTextObj 396

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 401) use a specify a filename.
Stream	This is a reference to an IStream or IWPStream 396 interface.

Note: When working with a placeholder object which embeds data into a PDF file (See

InputEmbeddedData 236) 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 150

8.26.2.18 IWPTextObj.MakeEndTag

Applies to
IWPTextObj 396

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 241 .
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.

8.26.2.19 IWPTextObj.MoveCursor

Applies to
IWPTextObj 396

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

8.26.2.20 IWPTextObj.MoveToPage

Applies to
IWPTextObj 396

Declaration
```
function MoveToPage(PageNr: Integer; X: Integer; Y: Integer): WordBool;
```

Description
not used yet.

8.26.2.21 IWPTextObj.ObjCommand

Applies to
IWPTextObj 396

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.

8.26.2.22 IWPTextObj.ScaleSize

Applies to
IWPTextObj 396

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.

8.26.2.23 IWPTextObj.Select

Applies to
IWPTextObj 396

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.

8.26.2.24 IWPTextObj.SetContentsID

Used for image cloning

Applies to
IWPTextObj 396

Declaration
```
function SetContentsID(ID: Integer): WordBool;
```

Description
GetContentsID 403 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 149

8.26.2.25 IWPTextObj.SetEmbText

Applies to
IWPTextObj 396

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!

8.26.2.26 IWPTextObj.SetFieldProp

Applies to
IWPTextObj 396

Declaration
```
procedure SetFieldProp(Id: Integer; const Text: WideString);
```

8.26.2.27 IWPTextObj.SetProp

Applies to
IWPTextObj 396

Declaration
```
procedure SetProp(ID: Integer; const Value: WideString);
```

Description
Currently this 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 [wpobjPositionInMargin]

15 : Set the mode [wpobjPositionInMargin]
16 : Set the mode [wpobjPositionInMargin]
To set the flag use the value "1", to clear it use the value "0".

8.26.2.28 IWPTextObj.SetPtr

Applies to
IWPTextObj 396

Declaration
```
function SetPtr(ObjectPtr: Integer): WordBool;
```

Description
This methods attaches this interface to a different object instance. Use with care.

8.26.2.29 IWPTextObj.ShowHint

Displays a hint window near this object.

Applies to
IWPTextObj 396

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.

8.27 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 401	WriteData 410
SaveName 409	WriteString 410
SavePath 410	

8.27.1 Properties

8.27.1.1 SaveName

Applies to
IWPTextWriter 409

Declaration
```
string SaveName;
```

Description
The name of the file which is beeing written.

8.27.1.2 SavePath

Applies to
IWPTextWriter 409

Declaration
```
string SavePath;
```

Description
The path of the file which is being written.

8.27.2 Methods

8.27.2.1 IWPTextWriter.WriteData

Applies to
IWPTextWriter 409

Declaration
```
procedure WriteData(Data: LongWord; DataLen: Integer);
```

Description
Write data from a buffer.

8.27.2.2 IWPTextWriter.WriteString

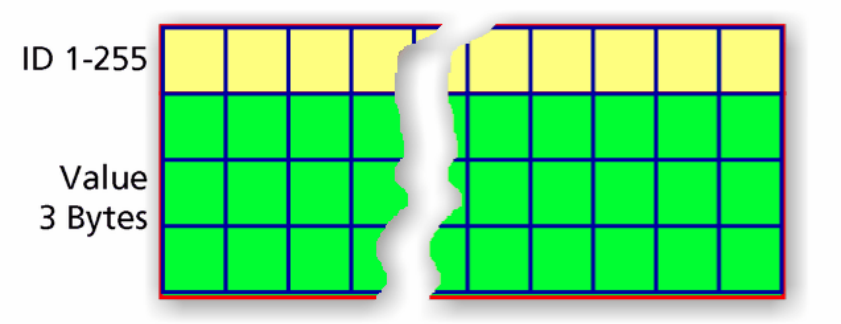
Applies to
IWPTextWriter 409

Declaration
```
procedure WriteString(const Text: WideString);
```

Description
Write a string to the output stream or file.

8.28 WPAT_codes

TextDynamic stores paragraph and paragraph style attribute values (such as indents and spacing) in arrays together with an id.



This makes it possible to not only store the value of a property but also to know that the propetrty 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 directy.

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.

IWPParInterface.ParAAddBits 360
IWPParInterface.ParAClear 360
IWPParInterface.ParADel 361
IWPParInterface.ParADelBits 361
IWPParInterface.ParAGet 361
IWPParInterface.ParAInc 361
IWPParInterface.ParASet 362
IWPParInterface.ParStrCommand 363

8.28.1 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 368 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 412 - see Convert Utility 349)
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 412 - see Convert Utility 349)
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 349
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!

8.28.2 Predefined Color Index Values

The methods CurrAttr.ColorToNr 280 and CurrAttr.NrToColor 286 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 349 functions in IWPParInterface 339.

8.28.3 Paragraph Attributes

```
// Margins
WPAT_IndentLeft = 17; // Indent Left  (CSS = margin)
WPAT_IndentRight = 18; // Indent Right (CSS = margin)
WPAT_IndentFirst = 19; // Indent First (CSS = text-indent)
WPAT_SpaceBefore = 20; // Space Before (CSS = margin)
WPAT_SpaceAfter = 21; // Space After  (CSS = margin)
WPAT_LineHeight = 22; // LineHeight in in % ( Has priority over WPAT_SpaceBetween)
WPAT_SpaceBetween = 23; // Space Between  (CSS = margin)  - negative : Absolute, Positive minimum

// padding, only in tables:
WPAT_PaddingLeft = 24; // Distance from Border to Text (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_WordSpacing = 28; // Value = 0 for default or % of EM (ignored for justifed text)
WPAT_Alignment = 29; // paralLeft, ...
WPAT_VertAlignment = 30; // Cells: paralVertTop, paralVertCenter, paralVertBottom

// Colors and background
WPAT_BGColor = 50; // Background Color - (as index value 412)  - see Convert Utility 349
WPAT_FGColor = 51; // Foreground Shading Color -  - see Convert Utility 349
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) 211 to set this flag according to the
used style names.
```

NOTE: gray symbols are reserved for future versions of TextDynamic!

8.28.4 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 349 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!

8.28.5 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 349
```
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 349
```

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
```

8.28.6 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 362.

8.29 WPA Actions

8.29.1 API

Predefined actions which are used by the toolbar

Description
In the PackageFile Manager you can click on "wpa Commands" to display a sorted table with the available action names. It can also automatically create C# template code or XML lines to be inserted in the "edit layout". This "edit layout" is used to create the toolbar and panels displayed by TextDynamic.

WPA actions can be used in your code as well. There are several methods to create a list of the available actions, to get the current, maybe localized, caption and hint and of course to execute the action.
Since all actions have an id we recommend to fill an array with the ids of the actions which are used by your application. Now you can use this index everywhere you need to executed or check an action.
The event OnUpdateGUI is very useful to update a custom menu or toolbar - you are not forced to use the one implemented inside of TextDynamic. Using the method wpaGetFlags you get the enabled/checked state of all actions in form of an array (OCX as WideString, .NET as Bytes[])

WPDLLInt.wpaCheck 120	This method is used to check the selected/hidden/disabled state for the action identified by the provided name.
WPDLLInt.wpaExec 120	Execute an action identified by an ID.
WPDLLInt.wpaGetFlags 120	This method retrieves the state of all WPA actions.
WPDLLInt.wpaProcess 122	
WPDLLInt.wpaSetFlags 122	This method can be used to update the flags provided by wpaGetFlags.

IWPMemo.wpaCheck 217
IWPMemo.wpaExec 217
IWPMemo.wpaGetCaption 217
IWPMemo.wpaGetID 218
IWPMemo.wpaProcess 218
IWPMemo.wpaSetCaption 218

8.29.2 Custom Actions

Since Version 1.3 of TextDynamic it is also possible to create custom action names in the PCC file.

The name of the Actions XML script must be "default" - only one is allowed in a PCC file.

Such Custom actions can be used to enable/disable buttons and menu entries in the GUI.

The action names can be used similar to predefined wpa actions in the toolbar description only that the name is specified in the "Action" instead of the "wpa" parameter.

Custom actions will get a dynamic action id which can be retrieved using method wpaGetID and used with wpaSetFlags / wpaGetFlags Parameters:

name: the name of the action
c: the default caption
h: the default hint
alias: optional alias string to be used by the application
param: optional string parameter

Custom action names can be used in the Action="" parameter for button and menu entries in the EditLayout.

Example:

```
<wpa>
  <action name="custom_example" c="Do something" h="This is just a test for custom
actions."/>
</wpa>
```

In the application wapSetFlags can be used to change the state of the custom action, ie. disable it:

```
private void WPDLLInt1_OnUpdateGUI(object Sender, int Editor,
        int UpdateFlags, int StateFlags,
        int PageNr, int PageCount, int LineNr)
{
   byte[] flags = WPDLLInt1.wpaGetFlags(Editor);
   int i = WPDLLInt1.wpaGetID("custom_example");
   if (i>=0)
      {
         flags[i] = (byte)(flags[i] & 0xE); // delete bit 1 = disable!
         WPDLLInt1.wpaSetFlags(Editor, 0, flags.Length, flags);
      }
}
```

8.29.3 List

Name	Caption	Hint/Description
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	Bullets
Center	Center	Center
Close	Close	Close
Copy	Copy	Copy
CreateTable	Create Table	Create Table
Cut	Cut	Cut
DecIndent	Prior Level	Prior Level
CombineCell	Combine Cells	Combine Cells
DelRow	Delete Row	Delete Row
Exit	Exit	Exit
FitHeight	Fit to Pageheight	Fit to Pageheight
FitWidth	Fit to Pagewidth	Fit to Pagewidth
SelAll	Select All	Select All
Hidden	Hidden	Hidden
IncIndent	Next Level	Next Level
SplitCells	Split Cell	Split Cell
InsRow	Insert Row	Insert Row
Italic	Italic	Italic
Bold	Bold	Bold
Justified	Justified	Justified
Left	Left	Left
Protected	Protected	Protected
New	New	New
Norm	Normal	Normal
Numbers	Numbers	Numbers
Paste	Paste	Paste
Undo	Undo	Undo
Print	Print	Print
PriorPage	Prior Page	Prior Page
NextPage	Next Page	Next Page
Right	Right	Right
HideSelection	Hide Selection	Hide Selection
SelectColumn	Select Column	Select Column
SelectRow	Select Row	Select Row
FindNext	Find Next	Find Next
Spellcheck	Spellcheck	Spellcheck
Underline	Underlined	Underlined
Strikeout	Strike Out	Strike Out
Subscript	Subscript	Subscript
Superscript	Superscript	Superscript
InsCol	Insert Column	Insert Column
DelCol	Delete Column	Delete Column
DeleteText	Delete Text	Delete Text
Redo	Redo	Redo
InsertNumber	Insert Number	Insert Number
InsertField	Insert Field	Insert Field
IsOutline	Include in Outline	Include in Outline
OutlineUp	Outline Level Up	Outline Level Up
OutlineDown	Outline Level Down	Outline Level Down
EditHyperlink	Edit Hyperlink	Edit Hyperlink
SpellAsYouGo	Spell-As-You-Go	Spell-As-You-Go
StartThesaurus	Thesaurus	Thesaurus
FontSelection	Font Name	Font Name
FontSizeSelection	Font Size	Font Size
FontColorSelection	Font Color	Font Color
FontBKColorSelection	Character Background	Character Background
ParColorSelection	Paragraph Background	Paragraph Background
ParStyleSelection	Paragraph Style	Paragraph Style
ParProtect	Protect Paragraph	Protect Paragraph
ParKeep	Keep Paragraph together	Keep Paragraph together
SelectLanguage	SelectLanguage	Change GUI Language
DiaPrint	Print	Print using Dialog
DiaPrinterSetup	Printer Setup	Printer Setup
DiaFind	Find	Find

Name	Caption	Hint/Description
DiaReplace	Replace	You can pass the initial text as string parameter, or @SELECTED@ Replace
DiaOPEN	Open	You can pass the initial text as string parameter, or @SELECTED@ Open File
DiaSAVE	Save	Save
DiaSAVEAS	SaveAs	SaveAs
DiaPreview	Preview	Open the preview dialog
		You can pass the zoom setting. Possible values are: WIDTH, DOUBLE, 10, 25, 50, 100, 150, 200, 500. The default is "Full page"
DiaPageProp	Page Properties	Page Properties
DiaPagePropEx	Page Properties	Page Properties
DiaSectionProp	Section Properties	Section Properties
DiaParagraphProp	Paragraph Properties	Paragraph Properties
DiaTabstops	Tabstops	Tabstops
DiaBullet	Bullets	Bullets
DiaBulletOutlines	Bullets and Numbering	Bullets and Numbering
DiaParagraphBorder	Paragraph Borders	Paragraph Borders
DiaStyleSheet	Stylesheet	Stylesheet
DiaOneStyle	Current Style Properties	Current Style Properties
DiaSpellcheck	Spellcheck	Spellcheck
DiaSpellOptions	Spellcheck Options	Spellcheck Options
DiaExportToWord	Export To Word	Export To Word
DiaExportToPDF	Export To PDF	Export To PDF
DiaINSSymbol	Insert Symbol	Insert Symbol
DiaINSTable	Insert Table	Insert Table
DiaINSGRAPHIC	Insert Graphic	Insert embedded Graphic
DiaINSGRAPHICLINK	Insert Graphic	Insert Graphic as link
DiaINSTextBox	Insert TextBox	Insert Textbox
DiaINSHyperlink	Insert Hyperlink	Insert Hyperlink
DiaINSBookmark	Insert Bookmark	Insert Bookmark
DiaINSFields	Insert Fields	Insert Fields
DiaReportBands	Report Properties	Report Properties
DiaPDFProperties	PDF Properties	PDF Properties
DiaPrintLabels	Print Labels	Print Labels
DiaPrintBooklet	Print Booklet	Print Booklet
DiaDocInfo	Document Info	Document Info
DiaTemplates	Edit Quick Templates	Edit Quick Templates
DiaDocVariable	Variable	Edit Document Variable
DiaWPAbout	About TextDynamic	About TextDynamic
DiaWPDebug	TextDynamic Debug	TextDynamic Debug
DiaMessageBox	Message	Message
DiaPagePropPaperTray	Paper Bins	Select Paperbins
DiaManageFormulas	Formulas	Manage Calculation Formulas
DiaManageHeaderFooter	Header/Footer	Manage Header/Footer Texts
DiaFontSelect	Font	Open a dialog to change font name and styles.
LayoutNormal	Normal	Normal Layout Mode
LayoutAsWebsite	As Website	Display as Website
LayoutAsOutline	As Outline	Display As Outline
zoom100	100%	Set Zoom to 100%
zoomwidth	Pagewidth	Set zoom to fit page width
zoomfullpage	Fullpage	Set zoom to fit full page
zoomdoublepage	Doublepage	Set zoom to fit 2 pages
zoomThumbnails	Thumbnails	Show Thumbnails
ShowCR	CR	Show Carriage Returns, Tabs and Newline Characters
ShowFields	Fields	Show mergefield markers
ShowFieldsEx	Fields	Show mergefield markers with names - see property Memo.ShowFields 174
ShowFormulas	Formulas	Show calculation names and formulas in tables
ShowBookmarks	Bookmarks	Show bookmark objects
ShowHyperlinks	Hyperlinks	Show hyperlink objects
ShowSPANCodes	SPAN	Show SPAN Code objects
ShowHiddenText	Hidden Text	Show hidden text
ShowWatermark 110	Watermark	Show watermark
ShowBackgroundImage 110	BackgroundImage	Show background pattern
ShowAddressArea	Address Area	Display frame where the address is
ShowFoldLine	Fold Line	
ShowInvalidateLastPage	Invalidate Last Page	Draw line at end of page
ShowInvalidateAllPages	Invalidate all Pages	
ShowPageMargins	Margins	Show page margins
ShowTableGrid	Grid	Draw invisible table borders
ShowHRuler	Ruler	Display/Hide the ruler

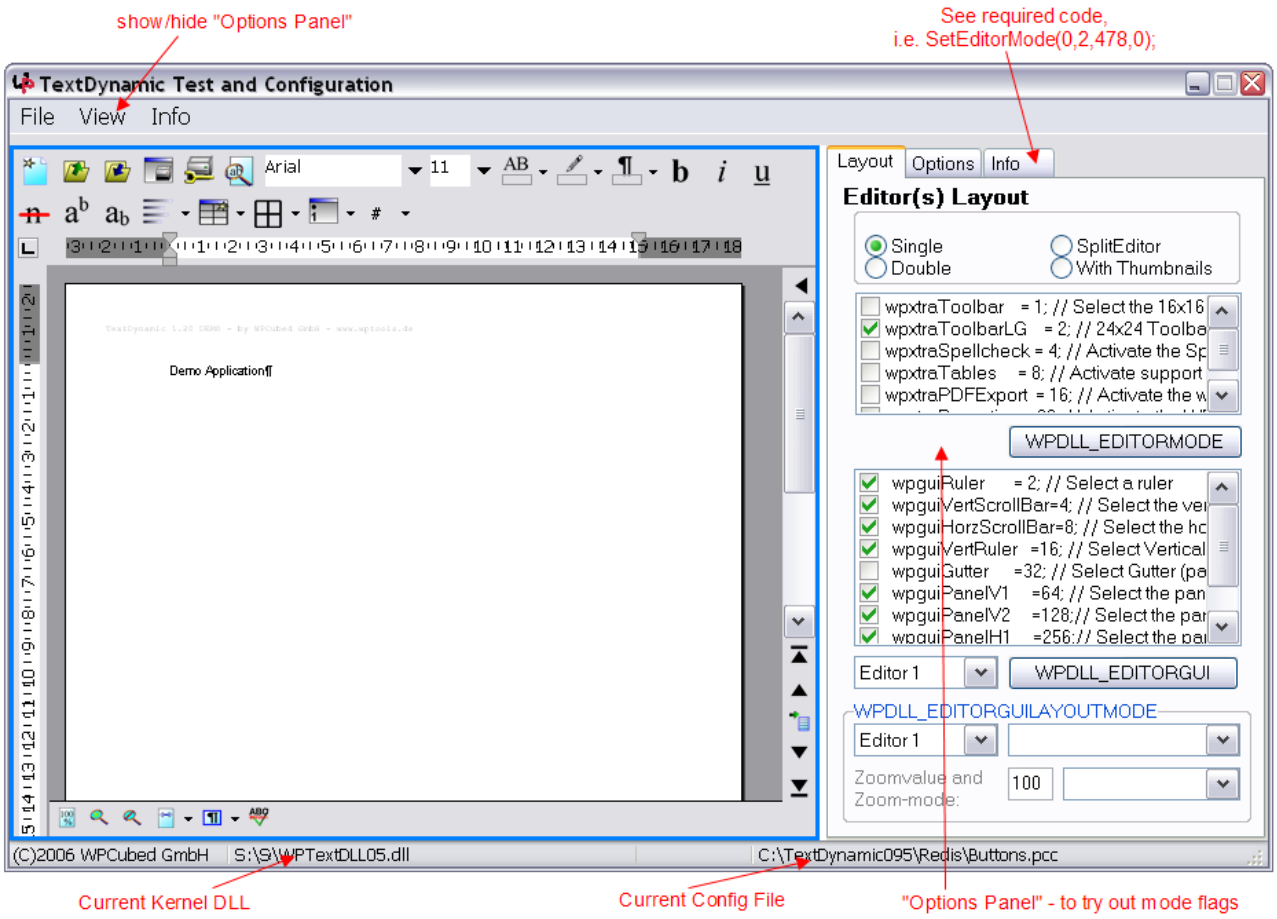
Name	Caption	Hint/Description
ShowVRuler	Vertical Ruler	Display/Hide the vertical ruler
ShowGutter	Gutter	Display/Hide the line number area
MoveFirst	First	First (Move command - See MoveSel)
MovePrior	Prior	Prior
MoveNext	Next	Next
MoveLast	Last	Last
MoveSel_PAGE	Move by Pages	Move by Pages
MoveSel_LASTSEARCH	Move by search string	Move by search string
MoveSel_SECTION	Locate Section	Locate Section
MoveSel_PAGEBREAK	Locate Pagebreaks	Locate Pagebreaks
MoveSel_TABLE	Locate Tables	Locate Tables
MoveSel_BOOKMARK	Locate Bookmarks	Locate Bookmarks
MoveSel_FIELD	Locate Fields	Locate Fields
MoveSel_HYPERLINK	Locate Hyperlinks	Locate Hyperlinks
MoveSel_IMAGE	Locate Images	Locate Images
MoveSel_HEADERS	Edit and Jump in all Headers	Edit and Jump in all Headers
MoveSel_FOOTERS	Edit and Jump in all Footers	Edit and Jump in all Footers
MoveSel_FOOTNOTE	Locate Footnotes	Locate Footnotes
Goto_MoveSel	Goto ...	Goto ...
Goto_Page	Goto Page #	Goto Page #
Goto_CPPOS	Goto Position	Goto Position
Goto_PARAGRAPH	Goto Paragraph Number	Goto Paragraph Number
Goto_LINE	Goto Line Number	Goto Line Number
Goto_TABLE	Goto Table Number or Name	Goto Table Number or Name
Goto_BOOKMARK	Goto Bokmark	Goto Bokmark
Goto_FIELD	Goto Field	Goto Field
Goto_HYPERLINK	Goto Hyperlink	Goto Hyperlink
Goto_Footnotes	Goto Footnote Nr.	Goto Footnote Nr.
Goto_BodyText	Goto Body Text	Leave Header or Footer area
NewHeader	Create Header for all Pages	create a header for all pages
NewFooter	Create Footer for all Pages	create a footer for all pages
InsertFootnote	Insert Footnote	Insert Footnote
InsertTextBox	Insert Textbox	Insert Textbox
InsertText	Insert Text	Insert Text
InsertField	Insert Field	Insert a defined field
InsertDocField	Insert Document Field	Insert a field from the document fields
InsertIniField	Insert Field	Insert Field
InsertTextFieldPAGE	Insert Page Number	Insert Page Number
InsertTextFieldNEXTPAGE	Insert Page Number of next Page	
InsertTextFieldPRIORPAGE	Insert Page Number of last Page	Insert Page Number of last Page
InsertTextFieldNUMPAGES	Insert Page Count	Insert Page Count
InsertTextFieldSECTIONPAGES	Insert Date (updated)	
InsertTextFieldDATE	Insert Time (updated)	Insert Time (updated)
InsertTextFieldTIME	Insert page number in Section	Insert page number in Section
InsertDATE	Insert current Date	Insert current Date
InsertTIME	Insert current Time	Insert current Time
InsertTextField	Insert defined Field	Insert defined Field
InsertTextCalcField	Insert calculation Field	Insert calculation Field
InsColBefore	Insert Column	Insert Column before current
InsRowBefore	Insert Row	Insert Row before current
SortRows	Sort Rows	Sort rows
AutoWidthCol	Autowidth	Auto resize column
TableToText	Table To Text	Convert table to text
TextToTable	Text To Table	Convert text to table
CombineTables	Combine Tables	Combine Tables
SplitTable	Split Table	Split Table
InsertToc	Insert TOC	Insert a Table of Contents (TOC)
IsHeadline	Mark as Headline	Mark this paragraph to be included in the TOC
UpdateToc	Update TOC	Update the Table of Contents
MergeBodyText	Merge Text	Merges fields in the body text
MergeAllTexts	Merge Text	Merges fields in the body and header and footer texts
ReformatAll	Reformat	Reformat
DeleteTrailingSpace	Delete Trailing Space	Delete empty space at the end
DeleteEmptyLines	Delete Empty Lines	Delete paragraphs with contain only empty mergefields
DeleteAllFields	Delete all Fields	Deletes all merge fields but keeps text
NormalizeText	Normalize Text	Remove all attributes from the text
CopyAttr	Copy Attributes	Copy the attributes of the current text
PasteAttr	Paste Attributes	Apply copied attributes to this text
FormatForScreen	Format For Screen	Optimized Formatting for Screeen
FormatDontBreakTables	Dont break Tables	Do not allow page breaks in tables

Name	Caption	Hint/Description
FormatDontBreakTableRows	Dont break Table Rows	Do not allow page breaks in table rows
FormatIgnoreKeepNInformation	Dont use Paragraph flow control	Ignore Keep and KeepN commands
FormatOrphanWidowControl	Use Orphan and Widow Control	Use Orphan and Widow Control
FormatUseHyphenation	Use Hyphenation	Use Hyphenation (insert manual hyphenation with Ctrl+ "-")
GraphicAsChar	Graphic As Character	Graphic As Character
GraphicRelToPar	Graphic positioned Relative To Paragraph	
GraphicRelToPage	Graphic Relative To Page	Graphic Relative To Page
GraphicNoWrap	No Textwrapping	No Textwrapping
GraphicAutoWrap	Wrapping on wider side	Wrapping on wider side
GraphicBothWrap	Wrapping on both sides	Wrapping on both sides
GraphicUnderText	Position under the Text	Position under the Text
GraphicNoFrame	Dont draw Frame	Dont draw Frame
GraphicSingleFrame	Draw single Frame	Draw single Frame
GraphicWideFrame	Draw Think Frame	Draw Think Frame
GraphicSaveData	Save Contents	Save Contents
GraphicLoadData	Load Contents	Load Contents
DropDownLayoutModes	Layout Modes	Layout Modes
DropDownShowModes	Show Modes	Show Modes
DropDownFindModes	Find Modes	Find Modes
DropDownDialogs	Dialogs	Dialogs
DropDownGraphicTools	Graphic Tools	Graphic Tools
DropDownTableTools	Table Tools	Table Tools
DropDownSpellCheck	Spellcheck Options	Spellcheck Options
DropDownOptions	Options	Options
font	->FontSelection	alias entry
size	->FontSizeSelection	alias entry
fontsize	->FontSizeSelection	alias entry
color	->FontColorSelection	alias entry
fontcolor	->FontColorSelection	alias entry
bgcolor	->FontBKColorSelection	alias entry
fontbgcolor	->FontBKColorSelection	alias entry
parcolor	->ParColorSelection	alias entry
style	->ParStyleSelection	alias entry
Search	->DiaFind	alias entry
Open	->DiaOpen	alias entry
PrinterSetup	->DiaPrinterSetup	alias entry
Replace	->DiaReplace	alias entry
Save	->DiaSave	alias entry
PrintDialog		

If you use wpaProcess - please note that most actions do not require a parameter - simply pass an empty string. Booloean actions, such as BOuter can be used with the parameter 1 to set or 0 to unset the respective property, otherwise they invert the value of the property. List actions such as FontSize and Font require the size or name of the font.

9 Test Application

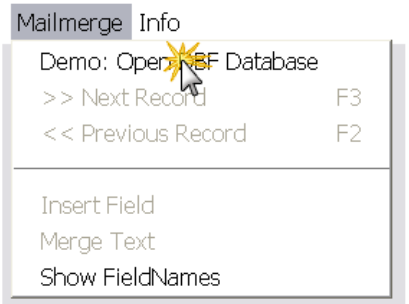
You can use the provided **test application** to test the different modes, flags and options.
You can apply an option to editor 1 and also to editor 2 or both editors at once.



You can also test how the mail merge can be used in TextDynamic.

We have integrated support for DBF data bases since this was possible without having to install external database drivers. Since the mail merge support uses events you can access any database which is supported by your development system and also insert calculated data.

In the demo however you can open a DBF file using the menu item "Open DBF Database"

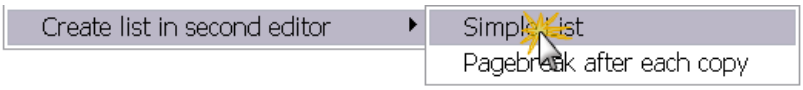


After the database was opened You can insert fields using the sub menu of the menu entry "Insert Field". Moving to next or previous record will also update the data in the text (you can press F2 and F3 to scroll through the database)

Example Form

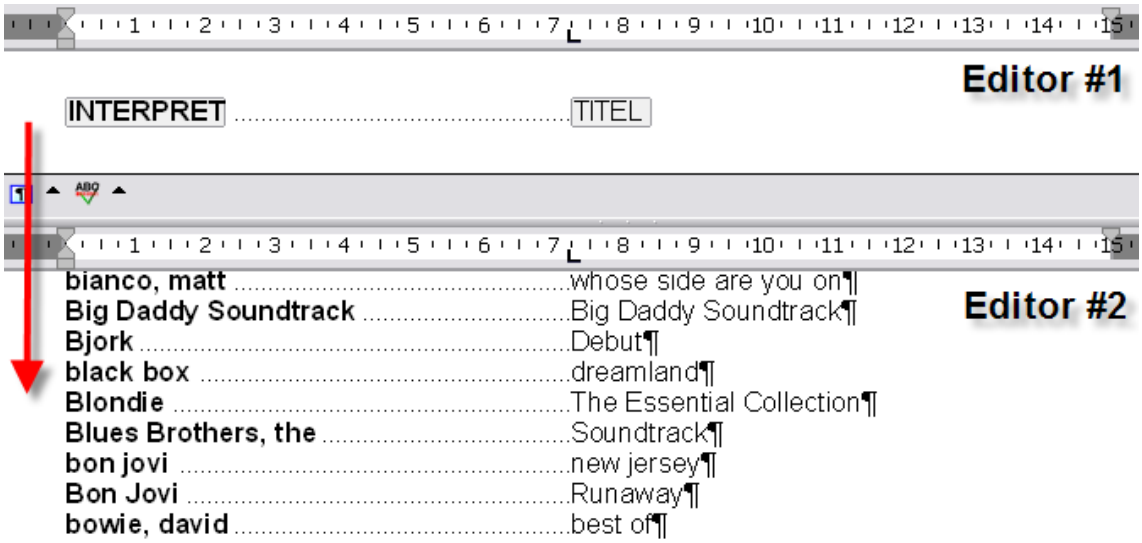
"Show Field-Names Mode" "Field Contents"

The demo also allows to create a list or multiple letters:



Example Form

The Template is in editor #1, the created list in editor #2:



The (pascal) code which does mail merge and creates the list:

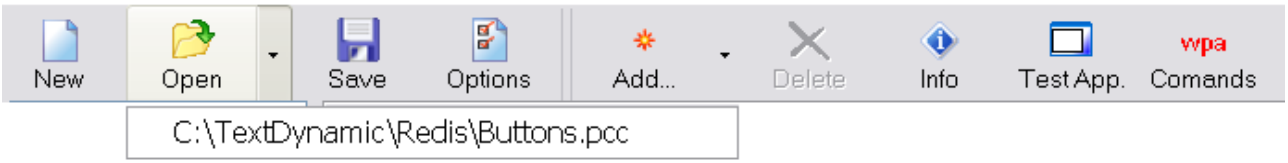
```
// Don't show the red field markers
       wp.Memo2.SpecialTextAttr(2).Hidden := TRUE;
// copy the template (styles, header + footer)
       wp.Memo2.LoadFromString(wp.Memo.SaveToString(false,'WPT'),false,'WPT');
// Clear the body text
       wp.Memo2.Clear(true, true);
// Move to first record
       Dbf1.First;
// Merge template in first editor for all records in table
       while not Dbf1.Eof do
       begin
         wp.Memo.MergeText('');
         // append the current text in editor 2
         if Sender=SimpleList1 then
               wp.Memo2.AppendOtherText(0)
         else  wp.Memo2.AppendOtherText(1);
         // Mode  0: simply append the other text.
         // Mode  1: create a new page and appends the text.
         // Mode  2: create a new page and section and appends the text.
         // Mode  3: appends the text as new section (also copies header+footer!)
         Dbf1.Next;
       end;
// Reformat the second editor and display it
       wp.Memo2.ReformatAll(false,true);
```

10 Package Files (Modify GUI)

Package files describe the graphical user interface (GUI) provided by the TextDynamic DLL.

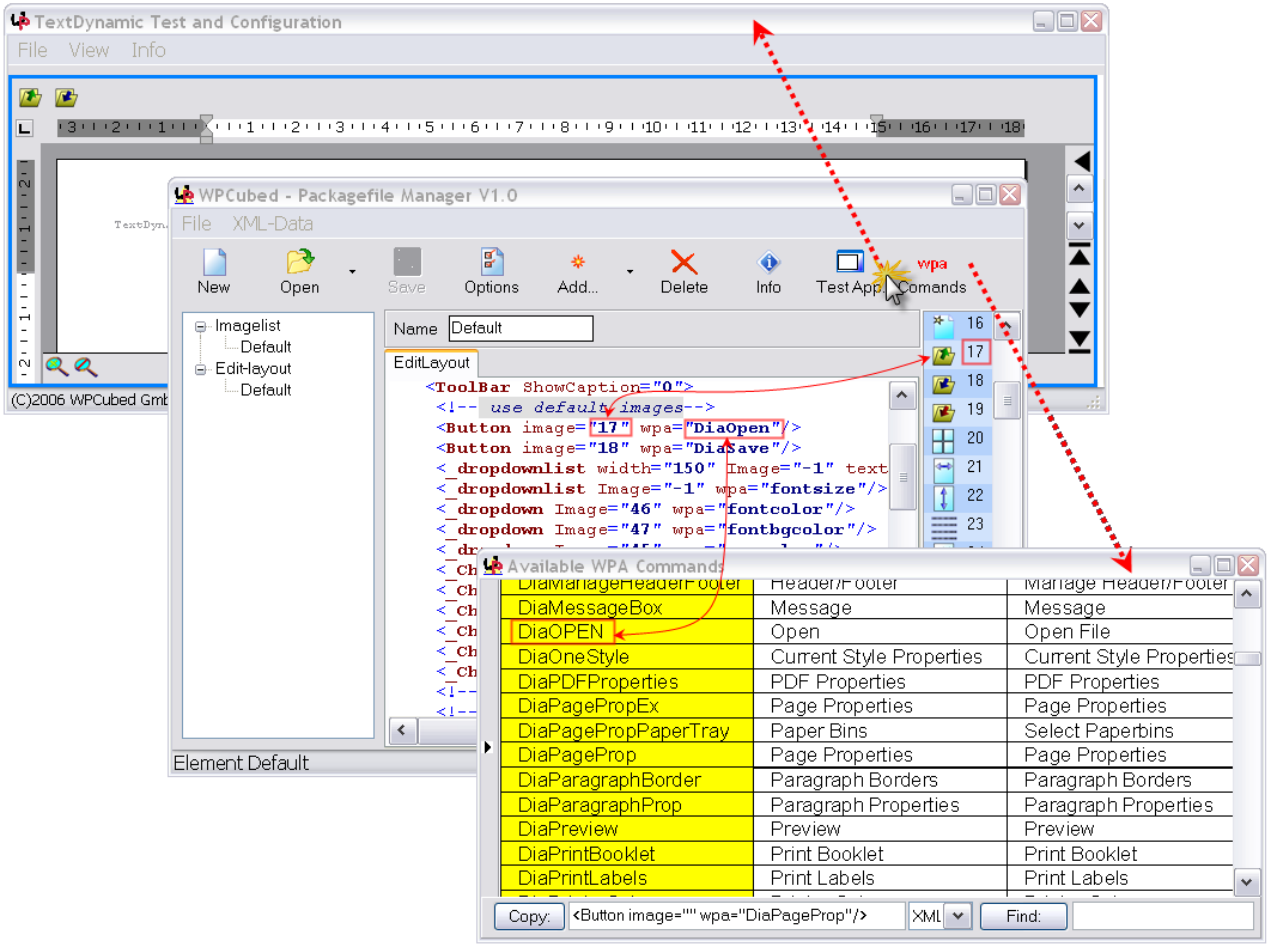
The application WPImagePack.exe is used to manage such a package file.

The menu of the package file manager:



New: Creates a new package file. It is not possible to override an existing file.
Open: Opens a package file. If encrypted a password dialog will appear. The dropdown shows the default PCC file.
Save: Save the current file at the known path. (Creates a backup of current file)
Options: Set options, such as password for the file.
Add: Create a new image repository (imagelist), layout or language XML script
Delete: Deletes current element
Info: Shows Application Infomessage
TestApp: Starts Testapplication or sends XML Data to Testapplication
wpa Commands: Shows List of possible wpa Commands

This screenshot shows how the last viewed image list (on right side), the *wpa* list, the test application and the XML editor can be used to configurate the toolbar(s). In this example only one 24x24 image list was ,loaded. We recommend to also use a 16x16 image list (with same name) for the images used by the small tool panels, such as the zoom panel in the lower left corner (PanelH1).



10.1 Working with image lists

a) to modify the images which are in the provided PCC file select the imagelist, click on "save" (only possible in registered version) and modify the created PNG file.

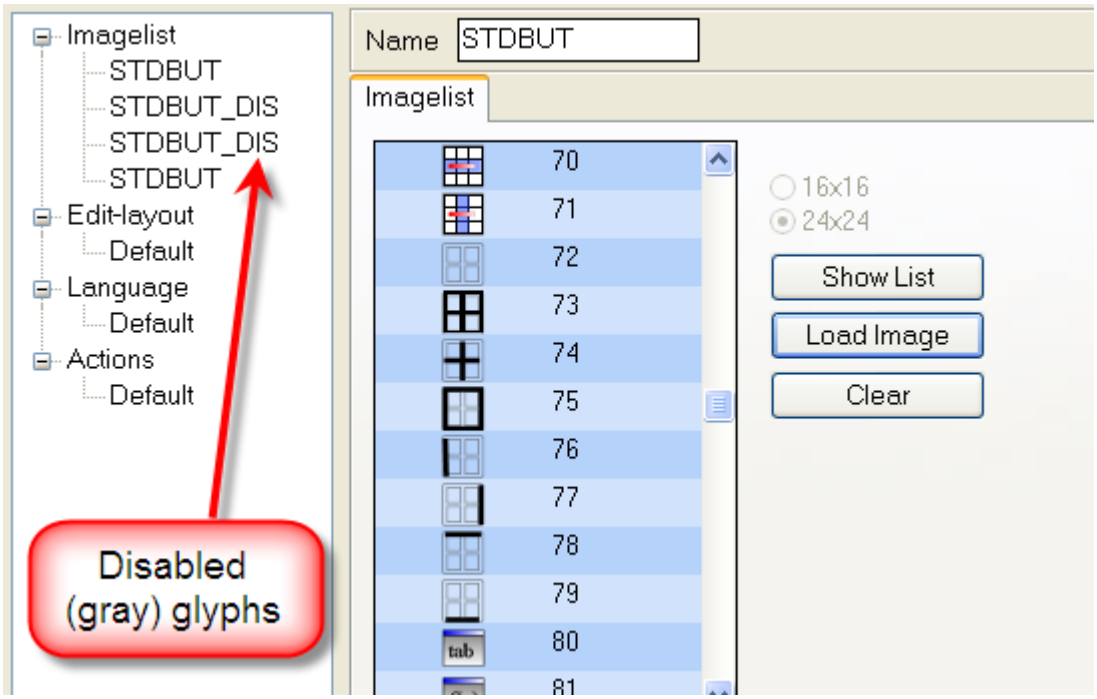
b) to create a new image list please create a new image list using the "New Element" button. Now type in a name, such as "STDBUT"

After pressing button "Clear" you can select the format of the image list, 16*16 or 24*24 pixels.

Now you can use the button "Load Image" to load a PNG file which contains the button images.

Please note:
- only PNG files can be loaded.
- the width of the PNG file must be the same as the selected format (16 or 24 pixels)
- the height must be N times the selected with
- It is not possible to append/exchange images lists. If you need to do this you will need to use the graphic application to create a new, higher PNG file. You can load this file after you have used the button "Clear". You can also save the current PNG data, modify it with a good graphics application and load it in. Please note, "MSPaint" is not capable to do it since it removes the transparency attribute. A recommended graphics application is available at www.pl32.com (English and German Version)

The DLL will select the images using the name and the size. It is best to provide 16*16 and 24*24 image lists under the same name - the engine will use the list which better matches the selected toolbar size.



The button "Show List" will open a non modal dialog which displays the current list with the image numbers. This widow will come in handy when you need to compare imagelists.

The EditLayout XML Editor will always display the image list which was opened last on the right hand side. This helps to use the right image index for the button XML tags.

10.2 Provided Glyphs

In the included PCC file we provide these glyphs - Copyright by WPCubed GmbH.

The TextDynamic developing license included the license to use these images in your Application.

Note: If you need additional Glyphs using the same style as these please let us know.

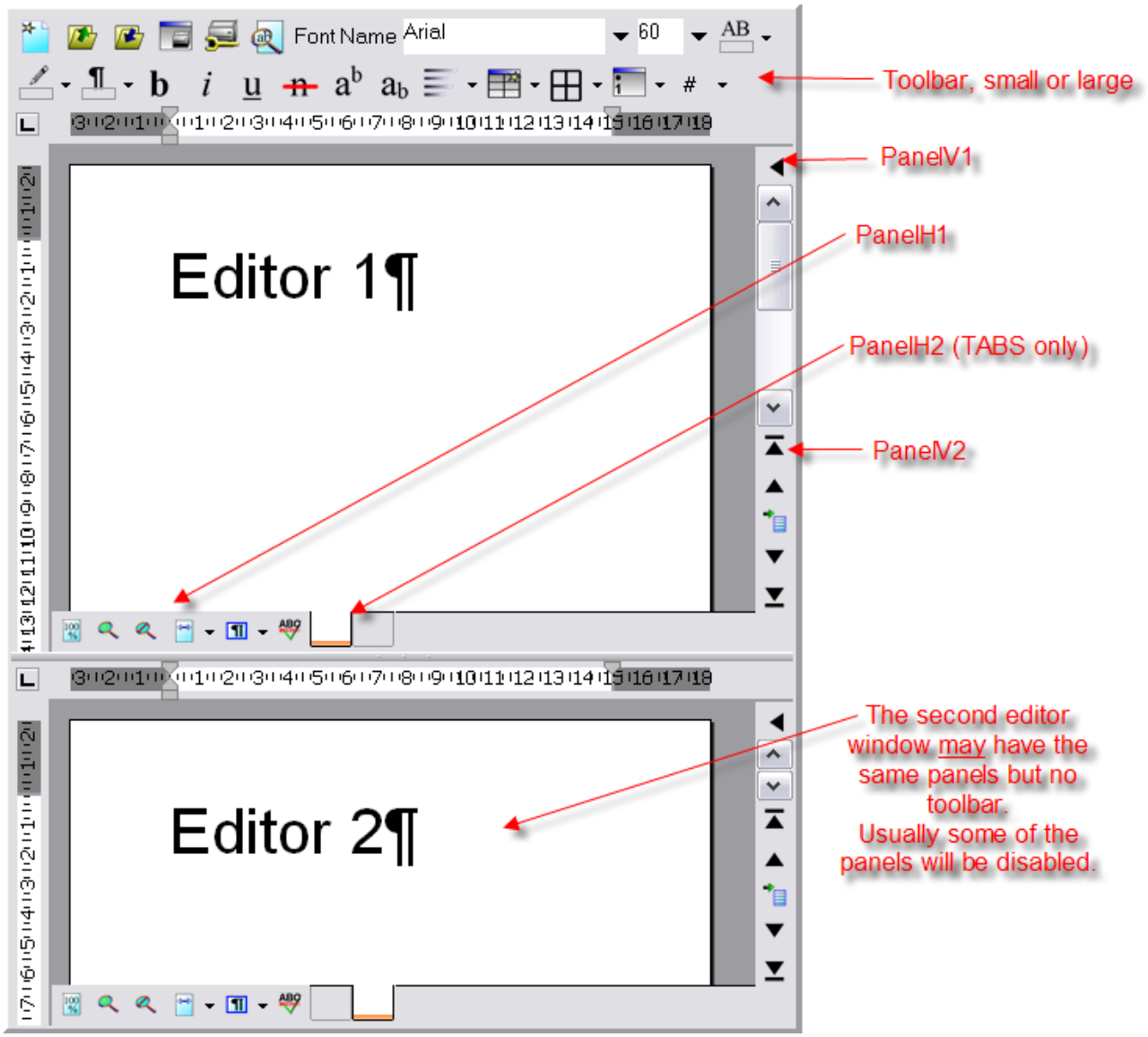
If you want to use different glyphs you can edit the PNG data included in the PCC file.

Of course you can also integrate images from a commercial glyph library such as GlyFX (http://www.glyfx.com)

10.3 Working with Edit-layouts

An *Edit-Layout* is the description of the toolbars shown by the TextDynamic DLL.

The following toolbars/panels can be used:



The PanelV2 is currently not used - it is reserved for next version.

This is the structure of the edit-layout XML information:

```
<?xml version="1.0" encoding="windows-1250"?>
<layout>
     <!-- global parameters, such as images and style, mode1, mode2 -->
     <main>
       <Toolbar>
             <!-- buttons ... -->
       </Toolbar>
       <PanelV1>
             <!-- buttons ... -->
       </PanelV2>
       <!-- other panels ... -->
     </main>
</layout>
```

Tip.: To debug an Edit-Layout XML script use the tag <debug> inside of <layout>. Debug messages will be sent to the debug console while the XML is interpreted.

10.3.1 <layout>

The outer tag <layout> always must exist.

The branch <main> is used to support different versions in the same XML file. Usually one branch <main> will be sufficient. Inside <main> the toolbar and the panels are described. The name "main" is used to select the definition in the API SetLayout().

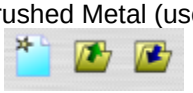
Other than the <main> branch the <layout> entry can contain the following XML tags:

<images> selects the default image lists for the toolbar and all panels. Each of the panel description can have its own <images> tag, too.

```
<images default="STDBUT" disabled="STDBUT_DIS"/>
```

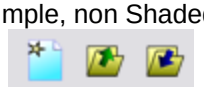
<design mode="0"**/>**
This modes are supported:
0= default - use global definition - use Command(9502,x) to change mode.
1= Titanium

2= Brushed Metal (use Command(9501, 1) to also let the forms use this background)



3= Shaded

This additional parameters can be used
colorfrom=#rrggbb"**,** default is $FFFFFF
colorto=#rrggbb"**,** default is $FE9696
4= Simple, non Shaded. Optionally the color can be selected with parameter "color"



You can use a **<design/>** tag inside of each panel description, for example <design mode=3/> to switch off the shading for a certain panel.

<style/> select the 3D appearance of the toolbars.

With btnframe=1 the "framed" mode is enabled :

```
<style btnframe="1" hoverframe="2"/>
```

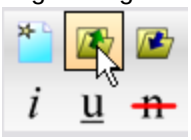
If parameter hoverframe=2 a frame will be drawn when the mouse is placed over the button. If btnframe=2 that frame will be always drawn.

```
<style btnframe="2" hoverframe="2"/>
```

If bit 3 is used in parameter btnframe the toolbar will use the 3D look known from older applications:

```
<style btnframe="6" hoverframe="2"/>
```

Default: if btnframe=0 or <style> is absent, the toolbar will use the modern hover mode (design can be changed using the tag <design> or, globally, with `wpdllInt1.Command(9502,x,0).`)

"Titanium"  "Simple" 

Note: The parameters are *btnframe* and *hoverframe* and can be also used in the <toolbar> and <Panel> tags.

Optionally:
<mode1 value=X> : The value X selects the editor layout, 0=single editor, 1=double editor, 2=split screen, 3=editor + thumbnails. The layout can be changed using an API, too.

<mode2 value=X> : The value X selected the editor mode. Bit 1 selects a small toolbar, bit 2 a large toolbar. Other bits select spllecheck, enabl tables and PDF export.

10.3.2 <toolbar> or <Panel..>

The <toolbar>, <panelH1>, <PanelH2> (reserved), <PanelV1> and <PanelV2> tags are placed in the branch <main>. They are used to define the respective toolbar or panel.

As parameter btnframe, hoverframe (see above), ShowCaption and color can be used. `<ToolBar Color="Red">` will create a completely red toolbar. This will only work if the global design has been set to "simple".

The parameter ShowCaption selects if the buttons should be displayed with a caption or not. 0 will disable the captions, 1 will enabled the caption. ShowCaption can be used with each button or group to override the setting made here.

The parameter columns=x can be used to limit the count of buttons displayed horizontally.

Inside the definition the following tags can be used:
<images/> - same use as in <layout>

<design/> - same use as in <layout>

<button> - define a simple push button.

The following parameters can be used:
Image=X X is the image number in the image list. The value -1 will disable the image.

Note: The EditLayout XML Editor will always display the image list which was opened last on the right hand side. This helps to use the right image index for the button XML tags.

ShowCaption=X Override the standard for this toolbar or group to display (X=1) or hide (X=0) the caption.

wpa=name This selects the action for this object, for example `wpa="Left"`. (See List 433) If you do not specify an action the event OnButtonClick will be triggered when the button was clicked. An action will also select a localizable caption and hint. Both can be overwritten using the following 2 parameters:

Caption=text Selects the caption for this button.

Hint=text Select the hint for this button

Font=fontname Select a certain font name, such as WingDings

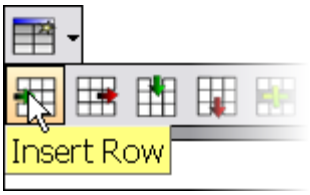
Param=text This is an optional string parameter - it can be used by the action or inside the OnButtonClick event.

IParam=X This is an optional integer parameter

Name=text The name of this element, optional for OnButtonClick event.

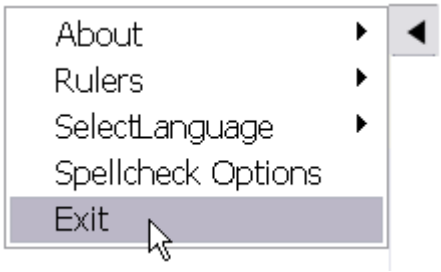
ifselected=no/yes Used for popup menu 437 items only. The menu is hidden if text is currently selected / not selected.

A button can also have *subitems*. In this case a drop down field will be displayed and the subitems will be displayed in a drop down panel. The parameter ShowCaption can also be used with the <subitems> tag.




```
<Button image="56" wpa="CreateTable">
    <subitems>
     <Button image="64" wpa="InsRow"/>
     <Button image="65" wpa="DelRow"/>
     ...
     </subitems>
</Button>
```

Alternatively a button can display menu when the user clicks. Menus are created using the nestable **<menu>** tags which use the same parameters as buttons (caption, wpa, ...).




```
<Button image="97">
 <menu caption="About" hint="This shows a popup menu!">
     <menu caption="PCC package file"
           wpa="DiaMessageBox" param="some text"/>
     <menu name="about" caption="This APP"/>
 </menu>
 <menu caption="Rulers">
 <menu wpa="SelectLanguage">
     <menu caption="German" wpa="SelectLanguage" param="DE"/>
     <menu caption="English" wpa="SelectLanguage" param="EN"/>
 </menu>
 <menu wpa="DiaSpellOptions"/>
 <menu name="exit" caption="Exit"/>
</Button>
```

<collection> - create button with sub elements. The image of the first selected subelement will be displayed.

Example, the paragraph alignment button:


```
<Collection>
    <subitems ShowCaption="1">
     <Radio Image="12" wpa="Left"/>
     <Radio Image="13" wpa="Center"/>
```

```
      <Radio Image="14" wpa="Justified"/>
      <Radio Image="15" wpa="Right"/>
      </subitems>
</Collection>
```

<check> - create a selectable button - uses the same parameters as <button>

<radio> - creates a selectable button. If selected all others in same group are deselected - uses the same parameters as <button>

<dropdown> - create a button with drop down menu. Use the parameter wpa=name to select the action, for example fontcolor, fontbgcolor or parcolor.

<dropdownlist> - create a combobox - Use the parameter wpa=name to select the action, for example font or fontsize.

<seperator> - seperator item

<text> - just a label. It can use the wpa parameter if the caption should be loaded from the action description.

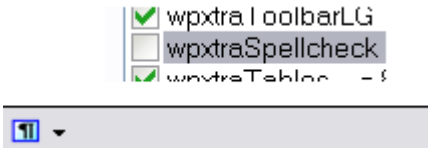
<group> - define a group of buttons - this is usefull with the <radio> objects.

To temporarily comment out a definition we recommend to preceed the tagname with a minus or underline character. The tag will not be recognized as mistake whe parsing the XML - but please don't forget to modify the closing tag as well.

Important:

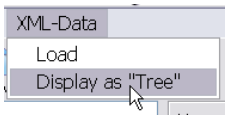
- Tags without contents should be inserted as closed tags: **<images .../>**
- Do not forget to close tags. If the toolbar does not show all buttons which were defined, it is likely that one button tag was not closed.
- If either your development license does not include a certain feature (i.e. PDF creation or spellcheck) **or** the feature has not been enabled using SetEditorMode(), the buttons which are connected to this feature will be automatically hidden:

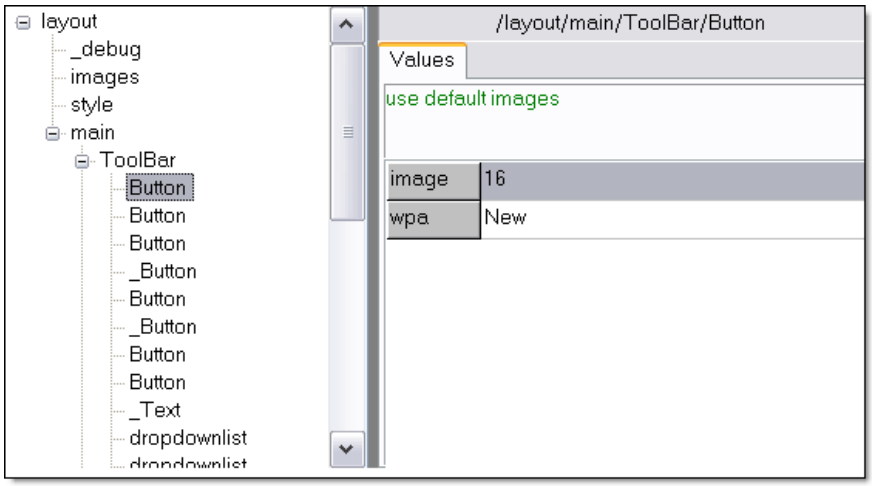
Spellcheck enabled:  and disabled:



10.3.2.1 Tree-Mode Editor

The Packagefile Manager can also display the XML data in tree:





This editor is useful if you only need to edit a certain parameter of one of the tags. You need to select the menu entry File\OK to save the file back to the main editor. The saving procedure will also format the file by indenting the XML tags. Please note that it only supports one comment for each tag, multiple comments will be discarded. So please use the Tree-mode editor carefully, we recommend to only use the main editor.

10.3.2.2 wpa Action Names

WPA Actions are typically assigned to the toolbar buttons and combo boxes.
They can also used with the method ProcessWPA(),
i.e. `WPDLLInt1.ProcessWPA "Size", "18.5"`

In the PackageFile Manager 427 you can click on 

to display a sorted table with the available names. It can also automatically create C# template code or XML lines to be inserted in the edit layout:

You only need to fill in the index of the image use the image list which is displayed on the right side of the editor to find the correct index.

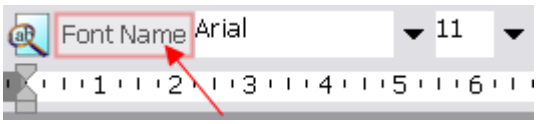
Tip: Use  to update the toolbar in the test application at once.

--> List of Action Names 419

10.3.3 Example

We want to demonstrate how easy it is to modify the toolbar.

If you do not like this text item in the default buttons.pcc file:



simply open the file buttons.pcc in the package file manager. Select the edit layout "Default" and scroll a bit down.

Under the XML tag **<toolbar ..>** you will see

EditLayout

```
<Button image="33" wpa="DiaPrinterSetup"/>
<Button image="28" wpa="DiaPreview"/>
<Text wpa="FontSelection"/>
<dropdownlist width="150" Image="-1" text="
<dropdownlist Image="-1" wpa="fontsize"/>
<!-- for colors use params: crx=2 crx1=18 c
     select the rectangle to show the curre
```

Now comment out the entry **<Text .../>** by simply adding a underscore _ before the XML tag.

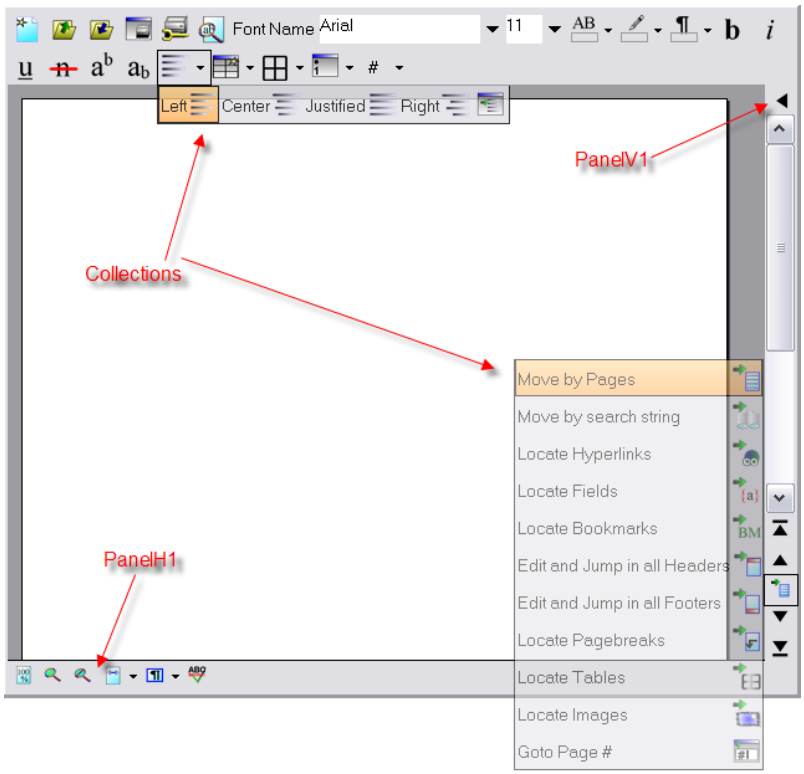
```
< _Text wpa="FontSelection"/>
```

Now save the PCC file. When you restart your application the text "Font Name" will be removed from the toolbar:

Note: You can use the button 

to transfer the XML data to the provided test EXE for immediate testing.

10.3.4 Template Layout




```
<?xml version="1.0" encoding="windows-1250"?>
<layout>
 <!-- Image selection for all toolbars which do not have an own <images/> tag-->
 <images default="STDBUT" disabled="STDBUT_DIS"/>
 <style btnframe="0" hoverframe="0"/>
```

```
<main>
<!-- This is the main toolbar ----------------------------
     force large mode with: large=1, small mode with large=0
     for standard flat buttons use: btnframe=1 hoverframe=2
     for old style buttons use: btnframe=6 hoverframe=2
     (can be set in "style")-->
 <ToolBar ShowCaption="0">
  <Button image="16" wpa="New"/>
  <Button image="17" wpa="DiaOpen"/>
  <Button image="18" wpa="DiaSave"/>
  <_Button image="18" wpa="Close"/>
  <Button image="31" wpa="DiaPrint"/>
  <_Button image="32" wpa="Print"/>
  <Button image="33" wpa="DiaPrinterSetup"/>
  <Button image="28" wpa="DiaPreview"/>
  <Text wpa="FontSelection"/>
  <dropdownlist width="150" Image="-1" text="Arial" wpa="font"/>
  <dropdownlist Image="-1" wpa="fontsize"/>
  <!-- for colors use params: crx=2 crx1=18 cry=15 cry1=19 to
       select the rectangle to show the current color.
       To specify the rectangle in 24X24 images use lcrc etc-->
  <dropdown Image="46" wpa="fontcolor"/>
  <dropdown Image="47" wpa="fontbgcolor"/>
  <dropdown Image="45" wpa="parcolor"/>
  <Check Image="1" wpa="Bold"/>
  <Check Image="2" wpa="Italic"/>
  <Check Image="6" wpa="Underline"/>
  <Check Image="5" wpa="Strikeout"/>
  <Check Image="3" wpa="SuperScript"/>
  <Check Image="4" wpa="SubScript"/>
  <!-- sub toolbar, must be inside of 'subitems'. It inherits the images !-->
  <Collection>
   <subitems ShowCaption="1">
    <Radio Image="12" wpa="Left"/>
    <Radio Image="13" wpa="Center"/>
    <Radio Image="14" wpa="Justified"/>
    <Radio Image="15" wpa="Right"/>
    <!-- Paragraph property dialog-->
    <Button Image="59" wpa="DiaParagraphProp" ShowCaption="0"/>
   </subitems>
  </Collection>
 <!-- Subtoolbar to create and manipulate table.-->
  <Button image="56" wpa="CreateTable">
   <subitems>
    <Button image="64" wpa="InsRow"/>
    <Button image="65" wpa="DelRow"/>
    <Button image="66" wpa="InsCol"/>
    <Button image="67" wpa="DelCol"/>
    <!-- split / combine horizontally-->
    <Button image="68" wpa="CombineCell"/>
    <Button image="70" wpa="SplitCells"/>
    <!-- Insert Table Dialog-->
    <Button Image="58" wpa="DiaINSTable"/>
   </subitems>
  </Button>
 <!-- sub toolbar for borders-->
  <dropdown image="73">
   <subitems>
    <Check Image="73" wpa="BAllOn"/>
    <Check Image="72" wpa="BAllOff"/>
    <Check Image="75" wpa="BOuter"/>
    <Check Image="74" wpa="BInner"/>
    <Check Image="76" wpa="BLeft"/>
    <Check Image="77" wpa="BRight"/>
    <Check Image="78" wpa="BTop"/>
    <Check Image="79" wpa="BBottom"/>
    <Button Image="58" wpa="DiaParagraphBorder"/>
   </subitems>
```

```
   </dropdown>
  <!-- numbering, bullets and indents-->
   <Button image="89" wpa="DiaBulletOutlines">
    <subitems>
     <check image="39" wpa="Numbers "/>
     <check image="38" wpa="Bullets"/>
     <Button image="41" wpa="IncIndent"/>
     <Button image="42" wpa="DecIndent"/>
     </subitems>
   </Button>
  <!-- Insert fields and symbols. Uses command WPDLL_COM_DATA_INPUT-->
   <dropdown image="-1" caption="#" ShowCaption="1">
    <subitems>
     <Button image="-1" wpa="InsertTextFieldPAGE" caption="#"/>
     <Button image="-1" wpa="InsertTextFieldNUMPAGES" caption="##"/>
    </subitems>
   </dropdown>
  </ToolBar>
<!-- The panel in the lower left corner-->
  <PanelH1 ShowCaption="0">
   <button image="24" wpa="zoom100"/>
   <button image="25" wpa="ZoomIn"/>
   <button image="26" wpa="ZoomOut"/>
   <Collection wpa="DropDownLayoutModes" image="31">
    <subitems>
     <button image="23" wpa="layoutnormal"/>
     <button image="21" wpa="zoomwidth"/>
     <button image="22" wpa="zoomfullpage"/>
     <button image="20" wpa="zoomdoublepage"/>
    </subitems>
   </Collection>
  <Collection wpa="DropDownShowModes" image="31">
    <subitems size="30">
     <check image="144" wpa="showcr"/>
     <check image="140" wpa="showTableGrid"/>
     <check image="142" wpa="showfields"/>
     <check image="141" wpa="showformulas"/>
     <check image="143" wpa="showbookmarks"/>
     <check image="146" wpa="ShowHyperlinks"/>
     <-check image="145" wpa="ShowSPANCodes"/>
     <check image="147" wpa="ShowHiddenText"/>
     <!-- some custom drawing-->
     <check image="136" wpa="ShowFoldLine"/>
     <check image="135" wpa="ShowAddressArea"/>
     <!-- toggle display of ruler etc-->
     <-check image="-1" wpa="ShowHRuler"/>
     <-check image="-1" wpa="ShowVRuler"/>
     <-check image="-1" wpa="ShowGutter"/>
    </subitems>
   </Collection>
  <check image="107" wpa="spellasyougo"/>
  </PanelH1>
<!-- The panel besides this one, reserved to be used as tabset only-->
<PanelH2>
</PanelH2>
<!-- cannot be modified -->
<!-- The Panel in the right upper corner, can be used for info menu-->
  <PanelV1 columns="1" ShowCaption="0">
   <Button image="97">
    <!-- create popup menu-->
    <menu caption="About" hint="This shows a popup menu!">
     <menu caption="PCC package file" wpa="DiaMessageBox"
             param="Loaded from file Buttons.PCC!"/>
     <menu name="about" caption="This APP"/>
     <-menu wpa="DiaWPAbout"/>
     <-menu wpa="DiaWPDebug"/>
     </menu>
   <menu caption="Rulers">
```

```
      <menu caption="Horizontal" wpa="ShowHRuler"/>
      <menu caption="Vertical" wpa="ShowVRuler"/>
      </menu>
    <menu wpa="SelectLanguage">
      <menu caption="German" wpa="SelectLanguage" param="DE"/>
      <menu caption="English" wpa="SelectLanguage" param="EN"/>
      </menu>
     <menu wpa="DiaSpellOptions"/>
    <menu name="exit" caption="Exit"/>
     </Button>
   </PanelV1>
 <!-- The panel in the lower right corner - for navigation-->
  <PanelV2 columns="1" showcaption="0">
   <Button image="90" wpa="movefirst"/>
   <Button image="91" wpa="moveprior"/>
   <Collection wpa="findmodes">
    <!-- inherit image from menu-->
    <subitems columns="4" size="30" showcaption="1">
     <radio Image="108" wpa="movesel_page"/>
     <radio Image="109" wpa="movesel_lastsearch"/>
     <radio Image="110" wpa="movesel_hyperlink"/>
     <radio Image="112" wpa="movesel_field"/>
     <radio Image="113" wpa="movesel_bookmark"/>
     <radio Image="114" wpa="movesel_headers"/>
     <radio Image="115" wpa="movesel_footers"/>
     <-radio Image="116" wpa="movesel_footnote"/>
     <radio Image="117" wpa="movesel_pagebreak"/>
     <radio Image="118" wpa="movesel_table"/>
     <radio Image="119" wpa="movesel_image"/>
     <-radio Image="117" wpa="movesel_section"/>
     <button Image="121" wpa="goto_page"/>
     <-button Image="121" wpa="goto_movesel"/>
    </subitems>
   </Collection>
   <Button image="93" wpa="movenext" hint="next"/>
   <Button image="94" wpa="movelast" hint="last"/>
   </PanelV2>
 </main>
</layout>
```

10.3.5 Popup Menues

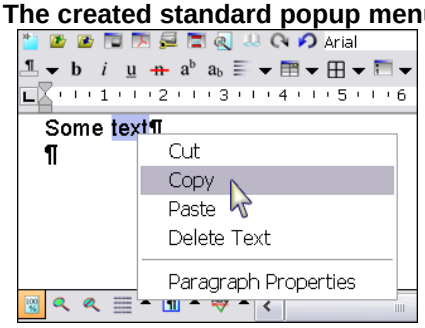
Popup menus can be specified inside the <main> tags as well.

Example:

```
<main>
  ...
  <!-- menues for the editor -->
  <menues>
    <!-- this context menu will be used for text -->
    <popup name="standard">
     <menu wpa="Cut"/>
     <menu menu wpa="Copy"/>
     <menu menu wpa="Paste"/>
     <menu menu wpa="DeleteText"/>
     <menu caption="-"/>
     <menu wpa="DiaParagraphProp"/>
    </popup>
    <!-- this context menu will be used for images -->
    <popup name="graphics">
      <menu wpa="GraphicAsChar"/>
      <menu wpa="GraphicRelToPar"/>
      <menu wpa="GraphicRelToPage"/>
      <menu wpa="GraphicBothWrap"/>
      <menu wpa="GraphicNoWrap"/>
    </popup>
```

```
  </menues>
</main>
```

The created standard popup menu:



In menus you can use the parameter ifselected:

ifselected="yes" will automatically hide a menu if there is no selection,
ifselected="no" will hide the menu if currently text is selected.

Tips:

1) You can use

```
<menu wpa="Bold" ifselected="yes"/>
<menu wpa="Italic" ifselected="yes"/>
<menu wpa="Underline" ifselected="yes"/>
```

in the standard menu to allow a possibility to toggle character styles.

2) If you want to allow pasting of simplified text (text attributes are removed) you can use the WPA actions PasteSimple and PasteText.

3) In case you need to temporarily disable a menu item please rename "menu" to "_menu" thus you do not need to create a HTML comment.

4) to create a sub menu You can nest the menu tags:

```
<menu caption="Character Attributes">
       <menu wpa="Bold"/>
       <menu wpa="Italic"/>
       <menu wpa="Underline"/>
       <menu wpa="StrikeOut"/>
       <menu caption="-"/>
       <menu wpa="SuperScript"/>
       <menu wpa="SubScript"/>
       <menu wpa="uppercase"/>
       <menu wpa="SmallCaps"/>
</menu>
```

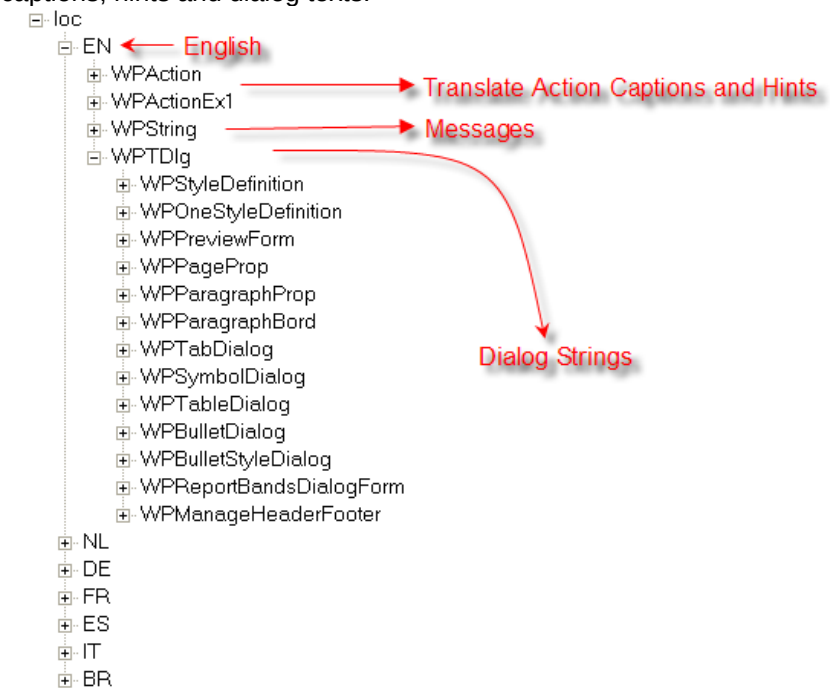
Please note the nesting of the tags: **<menu>...</menu>.**
The tag **<menu wpa="Bold"/>** has the "/" at the end to mark it as "closed" tag.

10.4 Language File

The Language description is an XML file similar to the Edit-Layout. Its name must be 'Default'.

The outer tag is **<loc>** and inside this tag several branches are possible, such as <EN>, <DE> or others. The name of such an branch is used by the SelectLanguage action to use the included strings to translate the

captions, hints and dialog texts.



This sub menu description will create two menu entries to select English or German:

```
<menu wpa="SelectLanguage">
     <menu caption="German" wpa="SelectLanguage" param="DE"/>
     <menu caption="English" wpa="SelectLanguage" param="EN"/>
</menu>
```

Of course the 'SelectLanguage' action can be executed directly using the ProcessWPA method,

VB Example:

```
WPDLLInt1.ProcessWPA "SelectLanguage", "DE"
```

11 TextDynamic Release Notes

20. January 2008 V1.42.1

- HTML reader created extra paragraph before table sometimes
- right click on misspelled word opened 2 popups
+ memo.DeleteParWithCondition 179 can now optionally just hide the paragraphs
+ new TextCommand 213 (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 139 to show a custom popup dialog please use Memo.SetBProp(0, 19, -1) 198 to deactivate the default popup dialog.
- IWPFieldContents.ContinueOptions 311 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 437 file:
  Example:
```
<menues>
     <popup name="standard">
       <menu wpa="Bold"/>
```
+ for popup menus 437 the parameter ifselected=yes/no has been added.

17. December 2007 V1.41

+ new TextCommand 208(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 198

30. November 2007 V1.40

+ added chapter in this manual Configure the Editor 51
* the chapter SetBProp 197 was updated.
* several improvements to editor
* fixed saving of spacing inside table cells
+ new PrintOption - print all colors in black (SetBProp 197)

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 233 and SetParName 261 can be used work with current table name.
+ ParStrCommand 363 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) 197. Can be combined with HTTP loading 211 option.

19. July 2007 - V 1.37

+ new method DrawToBitmap 124 (.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 426. (the images are included in file Buttons.pcc)
- bug fix in method GetPageAsMetafile 184
- bug fix for OnPaintWatermark 131 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 336 is automatically called when a section property is changed to select the modified value.
+ new "floppy" save button image in buttons.pcc
+ TextCursor.InputSection 247 can now select the current section to modify the current page attributes
+ Additions to TextCursor.CheckState 225
- Memo.EnumParagraphs did not use the EventParam parameter.

5. June 2007 - V1.35.6

+ RTFDataAppendTo 191 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 336 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 131 was updated and it now receives the page number in "Mode"
+ new ID in ParCommand 362 to check if it a paragraph is empty is empty
+ new ID in Memo.TextCommandStr 211 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 123)
+ new internal HTML syntax highlighter - activate it with Memo.TextCommand 208(12,1,0)
+ new source view (ie "HTML" source) - use Memo.TextCommandStr(5, 1, "HTML") 210

28. February 2007 - V1.31

+ New .NET method ReleaseInt 123

* the hyperlink detection now moves trailing dots after the link
+ The .NET method SetDLLName 123 will now pre-load the engine. This improves the speed when a lot editors are created dynamically.
+ New API: IWPTextCursor.TableSplit 267
IWPTextCursor.TableSort 267 sorts rows in a table
IWPTextCursor.ASetCellProp 268 changes attributes of a range of cells
IWPTextCursor.ASetCellStyle 269
IWPTextCursor.MergeCellHorz 270
IWPTextCursor.MergeCellVert 270
IWPTextCursor.ClearAllHeaders 271
IWPTextCursor.ClearAllFooters 271

9 February 2007 - V1.30.1
- SelectPrinter now works
- Custom Spellcheck 108 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 94
+ **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 418 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 162 is now also published by interface IWPMemo 160
+ new method MoveToField 256
+ new method InputEmbeddedData 236 - you can insert objects for PDF attachments (requires the optional PDF creation license)
+ new low level methods to move current paragraph (IWPParInterface.Select.. 368)
+ new convert methods 349 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 303, SelectNextPar 368)

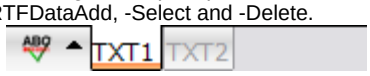
15 December 2006 - V1.29.3
- problem when using scroll wheel and pressing mouse button has been fixed.
+ automatic hyperlink detection now also works with email addresses

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 193 and Memo.GetPageAsMetafile 184
+ new property PageSizeList 167 to install a list of page size definitions (see demo "ExtractMetafiles 85")

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 79"
+ the spellcheck dialogs can be now localized suing the XML code in the PCC file
+ the **interface MAPI** 81 can be used to create e-mails (HTML incl. attachments) and send them
+ the **integrated MIME** 82 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 46" is now also available as VB.NET project

7 November 2006 - V1.28
+ **exciting new label feature** 76 (creation, preview and print)
+ Memo.RTFDataAppendTo 191 to create a large text from multiple copies of the current
+ Memo.SaveToVar 196 - save the text to a variant (.NET: "object") - for better performance
+ Memo.LoadFromVar 188 - 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 46. 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 62 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 143 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) 28"

26 October 2006 - V1.27.5
* new shaded look and other designs 58 for toolbars:  - reset with Command(9502,3,0)
+ Reporting interface 372 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** 15" - initialize the editor without a single line of code in VB6!
    * poperty **DLLName** 14 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 437 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 160.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 396.ShowHint - display hint at object
+ new: IWPTextObj.ObjType - read type of object

8 May 2006
+ easy creation of menu items (see C# demo 36)
- 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 63 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.