## Contents

### 1 How To Use the VPE Control

1.1 Introduction ................................................................. 1
1.2 VPE .NET Control .......................................................... 2
    1.2.1 Class Names and Namespaces ...................................... 3
    1.2.2 Exceptions .......................................................... 4
    1.2.3 Understanding the Data Types and Declarations ............... 4
    1.2.4 Visual Basic .NET .................................................... 5
    1.2.5 C# ................................................................. 7
1.3 VPE Java Control .......................................................... 8
    1.3.1 Exceptions .......................................................... 10
    1.3.2 Understanding the Data Types and Declarations ............... 10
1.4 VPE ActiveX ............................................................... 13
    1.4.1 Where are the constants like ‘VORIENT_PORTRAIT’ defined? 14
    1.4.2 Important Note for Users of Visual Basic ......................... 14
    1.4.3 Understanding the Data Types and Declarations (Visual Basic) 14
    1.4.4 Exceptions .......................................................... 16
    1.4.5 Live Demo on our Websites ........................................ 17
    1.4.6 CreateObject Call .................................................. 18
    1.4.7 Early Binding ....................................................... 18
    1.4.8 Visual FoxPro Note ................................................ 18
    1.4.9 MFC: Using the VPE ActiveX with the MFC without placing it in a Dialog Resource 19
    1.4.10 Using the VPE ActiveX in a CFormView derived class .......... 19
1.5 VPE VCL ................................................................. 20
    1.5.1 Understanding the Data Types and Declarations (Delphi) ...... 21
    1.5.2 Exceptions .......................................................... 22

### 2 Events Generated by the VPE - .NET Control

2.1 Events Generated by VPE .NET Control ................................ 25
2.2 AfterDestroyWindow Event .NET ........................................ 26
2.3 Closing Event .NET ........................................................ 27
2.4 AfterCloseWindow Event .NET .......................................... 28
2.5 BeforeOpenFile Event .NET .............................................. 29
2.6 AfterOpenFile Event .NET ............................................... 30
2.7 BeforeSaveFile Event .NET .............................................. 31
2.8 AfterSaveFile Event .NET ............................................... 32
2.9 HelpRequested Event .NET .............................................. 33
2.10 AfterAutoPageBreak Event .NET ...................................... 34
2.11 RequestPrint Event .NET ............................................... 36
2.12 BeforePrintNewPage Event .NET ...................................... 38
2.13 PrintDevData Event .NET ................................................ 40
2.14 BeforeMail Event .NET .................................................. 41
2.15 AfterMail Event .NET .................................................... 42
2.16 ObjectClicked Event .NET .............................................. 43
2.17 UDOPaint Event .NET .................................................... 44
2.18 AfterControlEnter .NET ................................................ 45
2.19 RequestControlExit .NET ............................................... 46
2.20 AfterControlExit .NET .................................................. 47
2.21 AfterControlChange .NET .............................................. 48
2.22 AfterFieldChange .NET ............................................... 49

### 3 Events Generated by the VPE - Java Control

3.1 Events Generated by VPE Java Control .................................. 51
3.2 AfterDestroyWindow Event Java ....................................... 54
3.3 Closing Event Java ........................................................ 55
4 Events Generated by VPE - ActiveX

4.1 Events Generated by VPE - ActiveX ............................................................... 79
4.2 AfterDestroyWindow Event ................................................................. 80
4.3 RequestClose Event ........................................................................... 81
4.4 AfterCloseWindow Event ................................................................. 82
4.5 BeforeOpenFile Event ........................................................................ 83
4.6 AfterOpenFile Event ........................................................................... 84
4.7 BeforeSaveFile Event ........................................................................ 85
4.8 AfterSaveFile Event ........................................................................... 86
4.9 DoHelpEvent ..................................................................................... 87
4.10 AfterAutoPageBreak Event ............................................................... 88
4.11 RequestPrint Event - ActiveX ............................................................ 89
4.12 BeforePrintNewPage Event ............................................................... 90
4.13 DoPrintDevDataEvent ...................................................................... 92
4.14 BeforeMail Event .............................................................................. 93
4.15 AfterMail Event ................................................................................. 94
4.16 DoObjectClickedEvent ...................................................................... 95
4.17 DoUDOPaintEvent .......................................................................... 96
4.18 AfterControlEnter ........................................................................... 97
4.19 RequestControlExit ......................................................................... 98
4.20 AfterControlExit ............................................................................. 99
4.21 AfterControlChange ...................................................................... 100
4.22 AfterFieldChange .......................................................................... 101

5 Events Generated by VPE - VCL

5.1 Events Generated by VPE - VCL ................................................................. 103
5.2 OnDestroyWindow Event .................................................................... 104
5.3 OnRequestClose Event ...................................................................... 105
5.4 OnCloseWindow Event ...................................................................... 106
5.5 OnBeforeOpenFileEvent ................................................................... 107
5.6 OnAfterOpenFileEvent ...................................................................... 108
5.7 OnBeforeSaveFileEvent ................................................................... 109
5.8 OnAfterSaveFileEvent ...................................................................... 110
5.9 OnHelpEvent ................................................................................... 111
5.10 OnAutoPageBreakEvent .................................................................. 112
5.11 OnRequestPrintEvent - VCL .............................................................. 114
5.12 OnPrintNewPageEvent .................................................................... 115
5.13 OnPrintDevDataEvent ..................................................................... 117
5.14 OnBeforeMailEvent ........................................................................ 118
5.15 OnAfterMailEvent .......................................................................... 119
5.16 OnObjectClickedEvent .................................................................... 120
5.17 OnUDOPaintEvent .......................................................................... 121
6 Management

6.1 Management ................................................. 127
6.2 OpenDoc .......................................................... 128
6.3 IsOpen .............................................................. 129
6.4 CloseDoc .......................................................... 130
6.5 SwapFileName .................................................... 132
6.6 EditProtection ..................................................... 134
6.7 License ............................................................. 135
6.8 EnableMultiThreading .......................................... 136
6.9 EnableURLs ....................................................... 137
6.10 LastError .......................................................... 138
6.11 Preview ............................................................ 143
6.12 PreviewDoc ....................................................... 144
6.13 CenterPreview .................................................. 145
6.14 BringPreviewToTop ............................................ 146
6.15 PreviewCtrl ...................................................... 147
6.16 ClosePreview .................................................... 148
6.17 IsPreviewVisible ............................................... 149
6.18 SetFocus .......................................................... 150
6.19 VpeSetFocus ...................................................... 151
6.20 Focus .............................................................. 152
6.21 VisualPage ....................................................... 153
6.22 DispatchAllMessages ......................................... 154
6.23 Refresh ........................................................... 155
6.24 DocExportType .................................................. 156
6.25 Compression ..................................................... 157
6.26 OpenFileDialog ................................................ 158
6.27 SaveFileDialog .................................................. 159
6.28 OpenFileName ................................................... 160
6.29 SaveFileName .................................................... 161
6.30 WriteDoc .......................................................... 162
6.31 WriteDocPageRange ............................................ 163
6.32 WriteDocStream ............................................... 164
6.33 WriteDocStreamPageRange .................................. 165
6.34 ReadDoc .......................................................... 166
6.35 ReadDocPageRange ............................................ 167
6.36 ReadDocStream ............................................... 168
6.37 ReadDocStreamPageRange .................................. 169
6.38 DocFileReadOnly ................................................ 170
6.39 ExternalWindow ................................................ 171
6.40 BorderStyle ..................................................... 172
6.41 Caption ........................................................... 173
6.42 Text .............................................................. 174
6.43 ToolBar ........................................................... 175
6.44 tbOpen ........................................................... 176
6.45 tbSave ........................................................... 177
6.46 tbPrint ........................................................... 178
6.47 tbPrintSetup ..................................................... 179
6.48 tbMail ............................................................ 180
6.49 EnableMailButton .............................................. 181
6.50 tbScale ........................................................... 182
6.51 tbGrid ............................................................ 183
6.52 GridMode ........................................................ 184
6.53 GridVisible ....................................................... 185
6.54 tbNavigate ....................................................... 186
6.55 tbHelp ........................................................... 187
9 Layout Functions

9.1 Layout Functions .................................................................................. 287
9.2 UnitTransformation ........................................................................... 288
9.3 EngineRenderMode ............................................................................. 289
9.4 PageBreak .......................................................................................... 290
9.5 AutoBreakMode ................................................................................. 291
9.6 PageCount .......................................................................................... 292
9.7 CurrentPage ......................................................................................... 293
9.8 PageFormat ......................................................................................... 294
9.9 PageWidth .......................................................................................... 301
9.10 PageHeight ......................................................................................... 302
9.11 PageOrientation ............................................................................... 303
9.12 PaperBin ........................................................................................... 305
9.13 StoreSet ............................................................................................ 307
9.14 UseSet ............................................................................................... 310
9.15 RemoveSet ......................................................................................... 311
9.16 nLeft, nTop, nRight, nBottom ............................................................. 312
9.17 nWidth, nHeight ............................................................................... 314
9.18 VFREE. ............................................................................................. 315
9.19 nFree ................................................................................................. 316
9.20 nLeftMargin, nTopMargin, nRightMargin, nBottomMargin .......... 317
9.21 SetDefOutRect .................................................................................. 318
9.22 SetOutRect ......................................................................................... 319
9.23 StorePos ............................................................................................ 320
9.24 RestorePos ......................................................................................... 321
9.25 Rotation ............................................................................................. 322
9.26 Viewable ............................................................................................ 323
9.27 Printable ............................................................................................ 324
9.28 ExportNonPrintableObject ................................................................ 325
9.29 Streamable ......................................................................................... 326
9.30 Shadowed .......................................................................................... 327
9.31 ClearPage .......................................................................................... 328
9.32 InsertPage .......................................................................................... 329
10 Rendering 335

10.1 Rendering .......................................................... 335
10.2 ComputeSingleLineChars ............................................. 336
10.3 nRenderWidth, nRenderHeight ..................................... 338
10.4 RenderPrint ......................................................... 339
10.5 RenderPrintBox ...................................................... 340
10.6 RenderWrite .......................................................... 341
10.7 RenderWriteBox ..................................................... 342
10.8 FontAscent ............................................................. 343
10.9 FontDescent ............................................................ 344
10.10 GetCharacterHeight .................................................. 345
10.11 FontInternalLeading .................................................. 347
10.12 FontExternalLeading .................................................. 348
10.13 RenderPicture ....................................................... 349
10.14 RenderPictureStream ................................................ 351
10.15 RenderPictureResID ................................................ 352
10.16 RenderPictureResName ............................................. 353
10.17 RenderPictureDIB .................................................... 354
10.18 RenderRTF ............................................................ 355
10.19 RenderBoxRTF ....................................................... 356
10.20 RenderRTFFile ....................................................... 357
10.21 RenderBoxRTFFile ................................................... 358
10.22 RenderRTFStream ................................................... 359
10.23 RenderBoxRTFStream ................................................ 360
10.24 RenderFormField ................................................... 361

11 Drawing Functions 363

11.1 Drawing Functions .................................................. 363
11.2 SetPen ............................................................... 364
11.3 NoPen ................................................................. 365
11.4 PenSize ............................................................... 366
11.5 PenStyle ............................................................. 367
11.6 PenColor ............................................................. 368
11.7 Line ................................................................. 369
11.8 VpeLine .............................................................. 370
11.9 PolyLine ............................................................. 371
11.10 AddPolyPoint ....................................................... 372
11.11 TVPEPolyLine.AddPoint .......................................... 373
11.12 BkgMode ............................................................ 374
11.13 BkgColor ............................................................ 375
11.14 BkgGradientStartColor ........................................... 376
11.15 BkgGradientEndColor ............................................. 377
11.16 BkgGradientTriColor ............................................. 378
11.17 BkgGradientMiddleColorPosition ................................ 379
11.18 BkgGradientMiddleColor ......................................... 380
11.19 BkgGradientRotation ............................................. 381
11.20 BkgGradientPrint .................................................. 382
11.21 BkgGradientPrintSolidColor ..................................... 384
11.22 TransparentMode .................................................. 385
11.23 HatchStyle ........................................................ 386
11.24 HatchColor ........................................................ 387
11.25 CornerRadius ...................................................... 388
11.26 Box ................................................................. 389
11.27 Polygon ............................................................. 390
11.28 AddPolygonPoint .................................................. 391
12 Text Functions

12.1 Text Functions........................................................................................................ 395
12.2 SetFont.................................................................................................................... 396
12.3 SelectFont................................................................................................................ 397
12.4 FontName................................................................................................................ 398
12.5 FontSize.................................................................................................................. 399
12.6 SetFontSubstitution................................................................................................. 400
12.7 PurgeFontSubstitution............................................................................................. 401
12.8 CharSet................................................................................................................... 402
12.9 CharSet.................................................................................................................... 403
12.10 TextAlignment........................................................................................................ 404
12.11 TextBold.............................................................................................................. 405
12.12 TextUnderline.................................................................................................... 406
12.13 TextUnderlined..................................................................................................... 407
12.14 TextItalic............................................................................................................. 408
12.15 TextStrikeOut...................................................................................................... 409
12.16 TextColor............................................................................................................ 410
12.17 Write.................................................................................................................... 411
12.18 VpeWrite............................................................................................................ 412
12.19 WriteBox............................................................................................................. 413
12.20 Print..................................................................................................................... 414
12.21 VpePrint............................................................................................................. 415
12.22 PrintBox............................................................................................................. 416
12.23 EmbeddedFlagParser......................................................................................... 417
12.24 DefineHeader..................................................................................................... 418
12.25 DefineFooter...................................................................................................... 419
12.26 CharPlacement.................................................................................................. 420

13 Text Block Object

13.1 CreateTextBlock.................................................................................................... 421
13.2 CreateTextBlockRTF............................................................................................ 422
13.3 Delete................................................................................................................. 423
13.4 HasText............................................................................................................... 424
13.5 Width.................................................................................................................. 425
13.6 Height............................................................................................................... 426
13.7 GetRangeHeight................................................................................................. 427
13.8 LineCount........................................................................................................... 428
13.9 WriteTextBlock................................................................................................... 429
13.10 RenderTextBlock............................................................................................... 430
13.11 Reset................................................................................................................. 431

14 Picture Functions

14.1 Picture Functions.................................................................................................. 432
14.2 PictureCacheSize................................................................................................. 433
14.3 PictureCacheUsed............................................................................................... 434
14.4 GetPictureTypes................................................................................................ 435
14.5 PictureType.......................................................................................................... 436
14.6 PictureCache....................................................................................................... 437
14.7 GetPicturePageCount........................................................................................ 438
14.8 PicturePage.......................................................................................................... 439
14.9 PictureEmbedInDoc............................................................................................. 440
14.10 PictureKeepAspectRatio................................................................................... 441
14.11 PictureBestFit.................................................................................................... 442
14.12 SetPictureDefaultDPI....................................................................................... 443
14.13 PictureX2YResolution...................................................................................... 444
14.14 PictureDrawExact............................................................................................. 445
14.15 PictureScale2Gray............................................................................................ 446
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.1</td>
<td>Barcode Functions (2D)</td>
<td>471</td>
</tr>
<tr>
<td>16.2</td>
<td>Bar2DAlignment</td>
<td>472</td>
</tr>
<tr>
<td>16.3</td>
<td>DataMatrixEncodingFormat</td>
<td>473</td>
</tr>
<tr>
<td>16.4</td>
<td>DataMatrixEccType</td>
<td>474</td>
</tr>
<tr>
<td>16.5</td>
<td>DataMatrixRows</td>
<td>475</td>
</tr>
<tr>
<td>16.6</td>
<td>DataMatrixColumns</td>
<td>476</td>
</tr>
<tr>
<td>16.7</td>
<td>DataMatrixMirror</td>
<td>477</td>
</tr>
<tr>
<td>16.8</td>
<td>DataMatrixBorder</td>
<td>478</td>
</tr>
<tr>
<td>16.9</td>
<td>DataMatrix</td>
<td>479</td>
</tr>
<tr>
<td>16.10</td>
<td>RenderDataMatrix</td>
<td>480</td>
</tr>
<tr>
<td>16.11</td>
<td>QRCODEVersion</td>
<td>481</td>
</tr>
<tr>
<td>16.12</td>
<td>QRCODEEccLevel</td>
<td>482</td>
</tr>
<tr>
<td>16.13</td>
<td>QRCODEMode</td>
<td>483</td>
</tr>
<tr>
<td>16.14</td>
<td>QRCODEBorder</td>
<td>484</td>
</tr>
<tr>
<td>16.15</td>
<td>QRCODE</td>
<td>485</td>
</tr>
<tr>
<td>16.16</td>
<td>RenderQRCODE</td>
<td>486</td>
</tr>
<tr>
<td>16.17</td>
<td>MaxiCode</td>
<td>487</td>
</tr>
<tr>
<td>16.18</td>
<td>RenderMaxiCode</td>
<td>488</td>
</tr>
<tr>
<td>16.19</td>
<td>MaxiCodeEx</td>
<td>489</td>
</tr>
<tr>
<td>16.20</td>
<td>RenderMaxiCodeEx</td>
<td>491</td>
</tr>
<tr>
<td>16.21</td>
<td>PDF417ErrorLevel</td>
<td>493</td>
</tr>
<tr>
<td>16.22</td>
<td>PDF417Rows</td>
<td>494</td>
</tr>
<tr>
<td>16.23</td>
<td>PDF417Columns</td>
<td>495</td>
</tr>
<tr>
<td>16.24</td>
<td>PDF417</td>
<td>496</td>
</tr>
<tr>
<td>16.25</td>
<td>RenderPDF417</td>
<td>497</td>
</tr>
<tr>
<td>16.26</td>
<td>AztecFlags</td>
<td>499</td>
</tr>
<tr>
<td>16.27</td>
<td>AztecControl</td>
<td>500</td>
</tr>
<tr>
<td>16.28</td>
<td>AztecMenu</td>
<td>501</td>
</tr>
<tr>
<td>16.29</td>
<td>AztecMultipleSymbols</td>
<td>502</td>
</tr>
<tr>
<td>16.30</td>
<td>AztecID</td>
<td>503</td>
</tr>
<tr>
<td>16.31</td>
<td>Aztec</td>
<td>504</td>
</tr>
<tr>
<td>16.32</td>
<td>RenderAztec</td>
<td>505</td>
</tr>
<tr>
<td>15.1</td>
<td>Barcode Functions</td>
<td>461</td>
</tr>
<tr>
<td>15.2</td>
<td>SetBarcodeParms</td>
<td>462</td>
</tr>
<tr>
<td>15.3</td>
<td>BarcodeMainTextParms</td>
<td>463</td>
</tr>
<tr>
<td>15.4</td>
<td>BarcodeAddTextParms</td>
<td>464</td>
</tr>
<tr>
<td>15.5</td>
<td>BarcodeAlignment</td>
<td>465</td>
</tr>
<tr>
<td>15.6</td>
<td>BarcodeAutoChecksum</td>
<td>466</td>
</tr>
<tr>
<td>15.7</td>
<td>BarcodeThinBar</td>
<td>467</td>
</tr>
<tr>
<td>15.8</td>
<td>BarcodeThickBar</td>
<td>468</td>
</tr>
<tr>
<td>15.9</td>
<td>Barcode</td>
<td>469</td>
</tr>
<tr>
<td>17.1</td>
<td>E-Mail Functions</td>
<td>507</td>
</tr>
<tr>
<td>17.2</td>
<td>Sending Mail on 64-Bit Windows</td>
<td>508</td>
</tr>
<tr>
<td>17.3</td>
<td>IsMAPIInstalled</td>
<td>509</td>
</tr>
<tr>
<td>17.4</td>
<td>MAPIType</td>
<td>511</td>
</tr>
<tr>
<td>17.5</td>
<td>MailSender</td>
<td>512</td>
</tr>
<tr>
<td>17.6</td>
<td>AddMailReceiver</td>
<td>513</td>
</tr>
<tr>
<td>17.7</td>
<td>ClearMailReceivers</td>
<td>515</td>
</tr>
<tr>
<td>17.8</td>
<td>AddMailAttachment</td>
<td>516</td>
</tr>
</tbody>
</table>
18 RTF Functions ................................................................. 525
  18.1 RTF Functions .................................................................. 525
  18.2 WriteRTF ........................................................................ 526
  18.3 WriteBoxRTF .................................................................... 528
  18.4 WriteRTFFile ................................................................... 529
  18.5 WriteBoxRTFFile .............................................................. 530
  18.6 WriteRTFStream ............................................................. 531
  18.7 WriteBoxRTFStream ......................................................... 532
  18.8 SetRTFFont ................................................................... 533
  18.9 SetRTFFont ................................................................... 534
  18.10 Build-In Paragraph Settings .......................................... 536
  18.11 FirstIndent .................................................................. 537
  18.12 LeftIndent .................................................................. 538
  18.13 RightIndent ................................................................ 539
  18.14 SpaceBefore ................................................................ 540
  18.15 SpaceAfter ................................................................ 541
  18.16 SpaceBetween .......................................................... 542
  18.17 DefaultTabSize .......................................................... 543
  18.18 SetTab ........................................................................ 544
  18.19 ClearTab ..................................................................... 545
  18.20 ClearAllTabs ............................................................... 546
  18.21 ResetParagraph .......................................................... 547
  18.22 Build-In Paragraph Settings: RTF Auto Page Break .......... 548
  18.23 KeepLines .................................................................. 549
  18.24 KeepNextParagraph .................................................... 550
  18.25 ParagraphControl ....................................................... 551

19 Clickable Objects ......................................................... 553
  19.1 Clickable Object .......................................................... 553
  19.2 EnableClickEvents ...................................................... 554
  19.3 ObjectID ..................................................................... 555
  19.4 ClickedObject ............................................................. 557

20 UDO - User Defined Objects ....................................... 558
  20.1 UDO - User Defined Objects .......................................... 558
  20.2 CreateUDO ................................................................. 559
  20.3 UDOPicture ................................................................. 560
  20.4 UDOPParam ................................................................. 561
  20.5 UDODC ...................................................................... 562
  20.6 UDOGraphics ............................................................... 564
  20.7 UDOIsPrinting ............................................................ 566
  20.8 UDOIsExporting .......................................................... 567
  20.9 UDODpiX .................................................................. 568
  20.10 UDODpiY ................................................................. 569
  20.11 nUDOLeft, nUDOTop, nUDORight, nUDOBottom ............... 570

21 Picture Export Functions ............................................. 571
  21.1 Picture Export Functions ................................................ 571
  21.2 JpegExportOptions ...................................................... 572
  21.3 TiffExportOptions ....................................................... 573
  21.4 BmpExportOptions ...................................................... 574
  21.5 PnmExportOptions ...................................................... 575
  21.6 GifExportOptions ........................................................ 576
22 Memory Streams (ActiveX / VCL) 587

22.1 Memory Streams (ActiveX / VCL) ................................................................. 587
22.2 CreateMemoryStream (ActiveX / VCL) ....................................................... 589
22.3 Close (ActiveX / VCL) .............................................................................. 590
22.4 Read (ActiveX / VCL) .............................................................................. 591
22.5 Write (ActiveX / VCL) .............................................................................. 592
22.6 Size (ActiveX / VCL) .............................................................................. 593
22.7 IsEof (ActiveX / VCL) ............................................................................. 594
22.8 State (ActiveX / VCL) ............................................................................. 595
22.9 Position (ActiveX / VCL) ...................................................................... 596
22.10 Seek (ActiveX / VCL) ........................................................................... 597
22.11 SeekEnd (ActiveX / VCL) .................................................................. 598
22.12 SeekRel (ActiveX / VCL) .................................................................... 599

23 Memory Streams 600

23.1 Memory Streams ...................................................................................... 600
23.2 CreateMemoryStream ............................................................................ 602
23.3 Close ......................................................................................................... 603
23.4 Read 604
23.5 Write ........................................................................................................ 605
23.6 WriteTo .................................................................................................... 606
23.7 Length ...................................................................................................... 607
23.8 IsEof608
23.9 State 609
23.10 Position .................................................................................................. 610
23.11 Seek (.NET) .......................................................................................... 611
23.12 Seek (Java, PHP, etc.) .......................................................................... 612
23.13 SeekEnd (Java, PHP, etc.) .................................................................... 613
23.14 SeekRel (Java, PHP, etc.) ................................................................. 614
23.15 SetLength (.NET) ............................................................................... 615
23.16 Flush (.NET) ....................................................................................... 616
23.17 CanRead (.NET) .................................................................................. 617
23.18 CanSeek (.NET) .................................................................................. 618
23.19 CanWrite (.NET) ................................................................................. 619

24 Charts 621

24.1 Charts .................................................................................................... 621
  24.1.1 The SmartChart Technology ................................................................. 623
  24.1.2 In VPE, Charts internally consist of two basic parts. ......................... 623
24.2 ChartDataCreate .................................................................................... 625
24.3 AddValue ............................................................................................... 626
24.4 AddLegend ............................................................................................. 627
24.5 XAxisTitle .............................................................................................. 628
24.6 YAxisTitle .............................................................................................. 629
24.7 AddXLabel ............................................................................................. 630
24.8 AddYLabel ............................................................................................. 632
24.9 SetColor .................................................................................................. 633
24.10 SetLineStyle ......................................................................................... 634
24.11 SetHatchStyle ..................................................................................... 635
24.12 SetPointSize ......................................................................................... 636
24.13 Minimum .............................................................................................. 637
24.14 Maximum ............................................................................................. 638
24.15 AddGap ................................................................................................. 639
24.16 AddRow ............................................................................................... 640
24.17 AddColumn........................................................................................................641
24.18 ChartDataAddValue..........................................................................................642
24.19 ChartDataAddLegend.........................................................................................643
24.20 ChartDataSetXAxisTitle.................................................................................644
24.21 ChartDataSetYAxisTitle.................................................................................645
24.22 ChartDataAddXLabel.......................................................................................646
24.23 ChartDataAddYLabel.......................................................................................648
24.24 ChartDataSetColor..........................................................................................649
24.25 ChartDataSetLineStyle....................................................................................650
24.26 ChartDataSetHatchStyle.................................................................................651
24.27 ChartDataSetPointType...................................................................................652
24.28 ChartDataSetMinimum....................................................................................653
24.29 ChartDataSetMaximum...................................................................................654
24.30 ChartDataAddGap..............................................................................................655
24.31 ChartDataAddRow.............................................................................................656
24.32 ChartDataAddColumn......................................................................................657
24.33 ChartTitle..........................................................................................................658
24.34 ChartTitleFontName........................................................................................659
24.35 ChartTitleFontSizeFactor................................................................................660
24.36 ChartSubTitle....................................................................................................661
24.37 ChartSubTitleFontSizeFactor..........................................................................662
24.38 ChartFootNote....................................................................................................663
24.39 ChartFootNoteFontName...................................................................................664
24.40 ChartFootNoteFontSizeFactor.........................................................................665
24.41 ChartAxesFontName..........................................................................................666
24.42 ChartAxisTitleFontSizeFactor.........................................................................667
24.43 ChartLegendFontName......................................................................................668
24.44 ChartLegendFontSizeFactor............................................................................669
24.45 ChartLineWidthFactor......................................................................................670
24.46 ChartBarWidthFactor.......................................................................................671
24.47 ChartRow...........................................................................................................672
24.48 ChartGridBkgColor..........................................................................................673
24.49 ChartGridBkgMode...........................................................................................674
24.50 ChartGridType....................................................................................................675
24.51 ChartGridColumn..............................................................................................676
24.52 ChartXGridStep..................................................................................................677
24.53 ChartYGridStep..................................................................................................678
24.54 SetChartYAutoGridStep...................................................................................679
24.55 ChartLegendPosition........................................................................................680
24.56 ChartLegendBorderStat...................................................................................681
24.57 ChartXLabelState..............................................................................................682
24.58 ChartPieLegendWithPercent..........................................................................683
24.59 ChartPieLabelType...........................................................................................684
24.60 ChartXLabelFontSizeFactor............................................................................685
24.61 ChartXLabelStep...............................................................................................686
24.62 ChartXLabelAngle............................................................................................687
24.63 ChartXLabelStartValue....................................................................................688
24.64 ChartYLabelState..............................................................................................689
24.65 ChartYLabelFontSizeFactor............................................................................690
24.66 ChartLabelsFontName.......................................................................................691
24.67 ChartYLabelStep...............................................................................................692
24.68 ChartYLabelDivisor.........................................................................................693
24.69 ChartGridRotation............................................................................................694
24.70 ChartYAxisAngle...............................................................................................695
24.71 ChartXAxisAngle...............................................................................................696
24.72 Chart..................................................................................................................697

25 FormFields........................................................................................................703
25.1 FormFields...........................................................................................................703
25.2 FormField.............................................................................................................704
25.3 CharCount............................................................................................................706
25.4 DividerPenSize....................................................................................................707
29 Field Object

29.1 Field Object - TVPEField ............................................................ 762
29.2 ObjectHandle ........................................................................... 763
29.3 IsNull ....................................................................................... 764
29.4 SetNull ..................................................................................... 765
29.5 NullValueText ......................................................................... 766
29.6AsString .................................................................................. 767
29.7 AsInteger .................................................................................. 768
29.8 AsNumber ............................................................................... 769
29.9 AsBoolean ............................................................................... 770
29.10 SetAsDateTime ......................................................................... 771
29.11 AsOleDateTime ......................................................................... 772
29.12 AsNetDateTime ......................................................................... 773
29.13 AsJavaDateTime ........................................................................ 774
29.14 Name ................................................................................... 775
29.15 Description ............................................................................ 776
29.16 DataSourceObject .................................................................... 777

30 Template Page Object

30.1 Template Page Object - TVPETemplatePage ............................ 778
30.2 ObjectHandle ........................................................................... 779
30.3 PageWidth ............................................................................... 780
30.4 PageHeight .............................................................................. 781
30.5 Orientation ............................................................................. 782
30.6 PaperBin ................................................................................... 783
30.7 nLeftMargin, nTopMargin, nRightMargin, nBottomMargin ......... 784
30.8 VpeObjectCount ................................................................. 785
30.9 VpeObject ............................................................................. 786

31 VPE Object

31.1 VPE Object - TVPEObject ...................................................... 787
31.2 ObjectHandle ........................................................................... 789
31.3 Kind 790
31.4 Name ................................................................................... 793
31.5 Text 794
31.6 ResolvedText .......................................................................... 795
31.7 nLeft, nTop, nRight, nBottom ..................................................... 796
31.8 PictureFileName ....................................................................... 797
31.9 TemplateObject ....................................................................... 798
31.10 InsertedVpeObjectCount ...................................................... 799
31.11 InsertedVpeObject ............................................................ 800
31.12 InsertedVpeObjectPageNo .................................................... 801
31.13 NextObject ........................................................................... 802

32 Interactive Objects

32.1 Interactive Objects ................................................................. 803
32.2 FormFieldControl ................................................................. 804
32.3 Checkbox ................................................................................. 806
32.4 RadioButtonGroup ............................................................... 807
32.5 RadioButton .......................................................................... 808
32.6 CheckmarkColor ................................................................. 809
32.7 DocContainsControls .......................................................... 810
32.8 ControlsModified ................................................................. 811
32.9 Interaction ............................................................................ 812
32.10 SetFocusToFirstControl ...................................................... 813
32.11 SetFocusControlByName ..................................................... 814
32.12 SetFocusControl ................................................................. 815
32.13 GetFocusControl ................................................................. 816
32.14 FindControl ......................................................................... 817
32.15 ControlEnabled ................................................................. 818
33 PDF Export ................................................................. 825
  33.1 PDF Export ............................................................... 825
  33.2 PDFVersion .............................................................. 826
  33.3 Author ................................................................. 827
  33.4 Title ................................................................. 828
  33.5 Subject ............................................................... 829
  33.6 Keywords ............................................................ 830
  33.7 Creator .............................................................. 831
  33.8 EmbedAllFonts ...................................................... 832
  33.9 DocExportPictureResolution .................................... 833
  33.10 DocExportPictureQuality ....................................... 834
  33.11 FastWebView ....................................................... 835
  33.12 UseTempFiles ...................................................... 836
  33.13 SubsetAllFonts .................................................... 837
  33.14 SetFontControl .................................................... 838
  33.15 ResetFontControl ................................................ 839
  33.16 Encryption ........................................................ 840
  33.17 EncryptionKeyLength ............................................ 841
  33.18 UserPassword ...................................................... 842
  33.19 OwnerPassword .................................................... 843
  33.20 Protection ........................................................ 844
  33.21 SetBookmarkDestination ........................................ 846
  33.22 BookmarkStyle .................................................... 848
  33.23 BookmarkColor .................................................... 849
  33.24 AddBookmark ....................................................... 850
  33.25 PDFALevel .......................................................... 852
  33.26 AddColorProfile .................................................. 853
  33.27 ExtIntDA ........................................................... 855

34 HTML Export ........................................................... 857
  34.1 HTML Export .......................................................... 857
  34.2 HtmlScale ............................................................ 858
  34.3 HtmlWordSpacing ................................................... 859
  34.4 HtmlRtfLineSpacing ................................................ 860
  34.5 HtmlCopyImages ..................................................... 861
  34.6 HtmlPageBorders .................................................... 862
  34.7 HtmlPageSeparators ............................................... 863

35 XML Export ............................................................. 865
  35.1 XML Export .......................................................... 865
  35.2 XMLPictureExport ................................................... 866

36 ODT Export ............................................................. 867
  36.1 ODT Export .......................................................... 867
  36.2 OdtTextWidthScale ................................................ 869
  36.3 OdtTextPtSizeScale ................................................ 870
  36.4 OdtLineHeight ....................................................... 871
  36.5 OdtAutoTextboxHeight ............................................ 872
  36.6 OdtPositionProtect ............................................... 873

37 Differences between the DLL and the ActiveX / VCL .................................................... 875
  37.1 Differences between the DLL and the Control ................................................ 875
1 How To Use the VPE Control

1.1 Introduction

The VPE Control is a wrapper around the VPE DLL (shared object / dylib).

It is available for the following programming languages:

- .NET
- Java
- PHP
- Python
- Ruby
- Delphi / C++ Builder VCL
- and as ActiveX

Because the events, properties and methods of the VPE Control are for all programming languages nearly 100% the same, we explain the VPE Control for all languages in this book.

This manual is the complete reference for all events, properties and methods provided by the VPE Control. It does not explain any programming techniques.

A first introduction on using VPE is given in the "Programmer's Manual" chapter 2 "Getting Started".

The techniques on using VPE are explained in the "Programmer's Manual" chapter 4 "Programming Techniques".

The installation procedure for each control is also described in detail in the "Programmer's Manual" chapter 1 "Installation".

Please read the following sections careful, which describe the basic techniques on using the VPE Control with each supported programming language. Then turn over to the "Programmer's Manual" and read the whole chapter 4 "Programming Techniques" very carefully. This document is a reference for the VPE-Control, not for Virtual Print Engine itself.
1.2 VPE .NET Control

The Winforms .NET Control, is based on the Winforms classes and should be used for GUI applications. It provides a preview window and related methods / properties.

VPE also includes a .NET WebServerControl (VpeWebControl) for use with ASP.NET. The control is a subset of the Winforms .NET Control. It does not have any events, properties or methods which are related to the graphical user interface (GUI) of VPE, since it is intended solely to be executed on servers within ASP.NET and therefore does not provide a GUI. All events, properties and methods not supported by the VpeWebControl are marked throughout this manual.

The common sequence of function calls is:

- Set the properties for behavior and appearance of VPE in the design mode or during runtime before calling the method OpenDoc()
- Open a document with the method "OpenDoc"
- Use all possible methods to insert VPE objects (like Write(), etc.)
- Use "PageBreak" to generate new pages
- Use "Preview" to show the preview to the user, this is optional
- Use "PrintDoc" to print the document, or WriteDoc() to export it
- Close the document with "CloseDoc"

It is only possible to open one document per VPE-Control. If you want to open multiple documents simultaneously, you need to place the same number of VPE-Controls on the form.

Example for Visual Basic .NET, which can be easily translated to other programming languages:

The following example assumes that you use a VPE .NET Control object named "Report".

```vbnet
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    Report.OpenDoc()
    Report.Print(1, 1, "Hello World!")
    Report.Preview()
End Sub
```

This example is very simple. The document is opened, the text "Hello World!" is inserted at position (1, 1) and afterwards the preview is shown.

For a detailed explanation of the basic programming techniques, please continue reading in the "Programmer's Manual" chapter 3 "Programming Techniques".
1.2.1 Class Names and Namespaces

As mentioned earlier, the VPE.NET Control is included in two different versions:

The **Winforms .NET Control** and the (WebForms) **.NET WebServerControl**, which is a subset of the Winforms .NET Control. The .NET WebServerControl does not have any events, properties or methods which are related to the graphical user interface of VPE, since it is intended solely to be executed on servers within ASP.NET and therefore does not provide a GUI. All events, properties and methods not supported by the VPE .NET WebServerControl are marked throughout this manual.

The Winforms .NET VPE Control has the class name *VpeControl*. It is derived from *System.Windows.Forms.UserControl*.

The VPE .NET WebServerControl has the class name *VpeWebControl*. It is derived from *System.Web.UI.Control*.

Depending on the edition of VPE you are using, the namespace of the control is:

<table>
<thead>
<tr>
<th>Edition</th>
<th>Winforms Namespace</th>
<th>WebServerControl Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Community</td>
<td>IDEALSoftware.VpeCommunity</td>
<td>IDEALSoftware.VpeWebCommunity</td>
</tr>
<tr>
<td>Standard</td>
<td>IDEALSoftware.VpeStandard</td>
<td>IDEALSoftware.VpeWebStandard</td>
</tr>
<tr>
<td>Enhanced</td>
<td>IDEALSoftware.VpeEnhanced</td>
<td>IDEALSoftware.VpeWebEnhanced</td>
</tr>
<tr>
<td>Professional</td>
<td>IDEALSoftware.VpeProfessional</td>
<td>IDEALSoftware.VpeWebProfessional</td>
</tr>
<tr>
<td>Enterprise</td>
<td>IDEALSoftware.VpeEnterprise</td>
<td>IDEALSoftware.VpeWebEnterprise</td>
</tr>
<tr>
<td>Interactive</td>
<td>IDEALSoftware.VpeInteractive</td>
<td>IDEALSoftware.VpeWebInteractive</td>
</tr>
</tbody>
</table>

You will notice that the class name of the VPE Control within the Winforms category is the same for each edition, i.e. it is *VpeControl* for the Standard Edition, as well as for the Enhanced Edition, the Professional Edition, etc. The same applies to the WebServerControl category, where the class name is *VpeWebControl* for all editions.

This makes it simple for you to migrate your source code from one edition of VPE to another, if you should decide later to upgrade to a higher edition. And since the VPE .NET WebServerControl constitutes a subset of the Winforms VPE Control, the migration of your source code from one platform to the other can be done within seconds.

**IMPORTANT NOTE:**

When using the *VpeWebControl*, always set the property *EnableMultiThreading* = true. The call-sequence must be:

```csharp
OpenDoc()
EnableMultiThreading = true
```
1.2.2 Exceptions
In case of error conditions, the .NET Control might throw exceptions. Properties and methods which throw exceptions are marked in this reference. VPE throws only one kind of exception: InvalidOperationException(). The informational message of the exception object explains in details the cause for each exception.

1.2.3 Understanding the Data Types and Declarations
Because the properties and methods of the VPE Control (for .NET, Java, PHP, Python, Ruby, ActiveX and Delphi / C++ Builder VCL) are nearly 100% the same, we explain the API of the VPE control for all the different programming languages within this single book. We use a meta-description to describe the data types, properties, methods and parameters of the VPE API.
1.2.4 Visual Basic .NET

Throughout this manual, methods and parameters are always declared in the form <Data Type> <Name>, e.g. boolean AutoDelete would be declared in VB as Dim AutoDelete as Boolean.

Data Types:

VpeCoord is Double
boolean is Boolean
integer is Integer
long is Integer
string is String
Color is Color

In addition there are defined many enumerations as data types for properties as well as for parameters of methods:

e.g. property PenStyle [integer] PenStyle

declares a property which has the enumeration type "PenStyle" and which is named "PenStyle". Data types in square-brackets, like "[integer]" in the example above, are not of interest for .NET users. They describe the data type for the ActiveX and VCL controls.

Example on using the above property: Doc.PenStyle = PenStyle.Solid

i.e. the property named PenStyle is assigned the value Solid of a pre-defined enumeration, which has the type PenStyle.

Methods:

The keyword method declares a method, e.g. method void CloseProgressBar().

A method is either a Sub or a Function, depending on how the return type of the method is declared. The keyword immediately following the word method declares the method’s return type. void is a Sub which does not return a value, e.g.

method void StorePos() is in Visual Basic: Sub StorePos()

all other declarations declare a function, e.g.

method boolean WriteDoc(string FileName) is in Visual Basic .NET
Function WriteDoc(FileName as String) as Boolean

Properties:

The keyword property describes a property. The line below the declaration of the property explains, how the property can be accessed (read-only, write-only or read / write together) and if it can be accessed during runtime only, or also during design time.

Example:

property boolean PageScrollerTracking

read / write; design- & runtime
means: the property is of type Boolean, its name is PageScrollerTracking and it can be accessed for read (you can read its value, e.g. n = PageScrollerTracking) and for write (you can assign it a value, e.g. PageScrollerTracking = True) and it is available during runtime (while your application is running) and design time (when you are developing your forms).

It would be declared in Visual Basic .NET as Dim PageScrollerTracking as Boolean
1.2.5 C#

Throughout this manual, methods and parameters are always declared in the form as it is done in C#. Only some data types are named differently.

Data Types:

VpeCoord is double
boolean is bool
integer is int
long is int
string is string
Color is Color

In addition there are defined many enumerations as data types for properties as well as for parameters of methods:

e.g. property PenStyle [integer] PenStyle
declares a property which has the enumeration type "PenStyle" and which is named "PenStyle". Data types in square-brackets, like "[integer]" in the example above, are not of interest for .NET users. They describe the data type for the ActiveX and VCL controls.

Example on using the above property: Doc.PenStyle = PenStyle.Solid
i.e. the property named PenStyle is assigned the value Solid of a pre-defined enumeration, which has the type PenStyle.

Methods:

The keyword method declares a method, e.g. method void CloseProgressBar().
Or method boolean WriteDoc(string FileName)

Properties:

The keyword property describes a property. The line below the declaration of the property explains, how the property can be accessed (read-only, write-only or read / write together) and if it can be accessed during runtime only, or also during design time.

Example:

property boolean PageScrollerTracking
read / write; design- & runtime

means: the property is of type bool, its name is PageScrollerTracking and it can be accessed for read (you can read its value, e.g. n = PageScrollerTracking) and for write (you can assign it a value, e.g. PageScrollerTracking = True) and it is available during runtime (while your application is running) and design time (when you are developing your forms).

It would be declared in C# as public bool PageScrollerTracking
1.3 VPE Java Control

The VPE Java Control is shipped in two different versions: the GUI and the Non-GUI version. The GUI version is only available for Windows and offers an optional preview window. The Non-GUI version has some special features on the Windows platform:

- Printing documents directly to printers
- Exporting pages as image files
- Mailing documents

The version on non-Windows platforms lacks the above features, but is in all other aspects 100% identical to the Windows-Version.

The GUI version can be found in the file <vpe-installation-directory> / deploy / vpegui.jar

The Non-GUI version is available for all platforms and can be found in the file <vpe-installation-directory> / deploy / vpe.jar

Make sure that any of the two VPE JAR files is in your classpath.

For the Non-GUI version the recommended way of using VPE is to export a created document to PDF and to use a PDF Reader (like Adobe Acrobat) for displaying and/or printing the document, or to send the PDF document via HTTP to a client browser.

For the use with server applications on the Windows platform, the Non-GUI version is recommended.

The common sequence of function calls is:

- Set the properties for behavior and appearance of VPE during runtime before calling the method openDoc()
- Open a document with the method "OpenDoc"
- Use all possible methods to insert VPE objects (like Write(), etc.)
- Use "PageBreak" to generate new pages
- Use "Preview" to show the preview to the user, this is optional and only supported in the GUI version for the Windows platform
- Use "PrintDoc" to print the document (only supported in the GUI version for the Windows platform), or WriteDoc() to export it
**Example:**

Create a file named “VpeTest.java” with the following code:

```java
import com.idealSoftware.vpe.*;
import com.idealSoftware.vpe.events.*;

class VpeTest {
    public static void main(String[] args) {
        VpeControl doc = new VpeControl();
        doc.openDoc();
        doc.print(1, 1, "Hello World!");
        doc.writeDoc("hello world.pdf");
        doc.closeDoc();
    }
}
```

Compile with: `javac -classpath ..\deploy\vpe.jar VpeTest.java`

Execute with: `java -cp .;../deploy/vpe.jar VpeTest`

Make sure to adjust the classpath, so that it points to the VPE JAR file in your environment.

For the Windows platform, here is a version which shows a preview window:

Please note that the major difference to the previous version is one additional line of code with a call to “Doc.preview()” (and a method `pause()`).

Create a file named “VpeGuiTest.java” with the following code:

```java
import com.idealSoftware.vpe.*;
import com.idealSoftware.vpe.events.*;
import java.io.*;

class VpeGuiTest {
    static void pause() {
        try {
            System.out.printf("\n\n  Press ENTER...");
            System.in.read();
            while (System.in.available() > 0)
                System.in.read(); // flush the buffer
        } catch (IOException e) {
            System.out.printf("Error
");
        }
    }

    public static void main(String[] args) {
        VpeControl doc = new VpeControl();
        doc.openDoc();
        doc.print(1, 1, "Hello World!");
        doc.preview();
        doc.writeDoc("hello world.pdf");
        pause();
        doc.closeDoc();
    }
}
```
Compile with: `javac -classpath ..\deploy\vpegui.jar VpeGuiTest.java`

Execute with: `java -cp .;../deploy/vpegui.jar VpeGuiTest`

Make sure to adjust the classpath, so that it points to the VPE JAR file in your environment.

This example is very simple. A document is created, the text "Hello World!" is inserted at position (1, 1) and afterwards the document is exported to the PDF file named “hello word.pdf”.

For a detailed explanation of the basic programming techniques, please continue reading in the "Programmer’s Manual" chapter 3 "Programming Techniques".

1.3.1 Exceptions

In case of error conditions, the Java Control might throw exceptions. Properties and methods which throw exceptions are marked in this reference. VPE throws only one kind of exception: `InvalidOperationException`. The informational message of the exception object explains in details the cause for each exception.

1.3.2 Understanding the Data Types and Declarations

Because the properties and methods of the VPE Control (for .NET, Java, PHP, Python, Ruby, ActiveX and Delphi / C++ Builder VCL) are nearly 100% the same, we explain the API of the VPE control for all the different programming languages within this single book. We use a meta-description to describe the data types, properties, methods and parameters of the VPE API.

Here is how the meta-description is converted to Java:

Throughout this manual, methods and parameters are always declared in the form as it is done in Java. Only some data types are named differently.

**Data Types:**

- `VpeCoord is double`
- `boolean is boolean`
- `integer is int`
- `long is long`
- `string is String`
- `Color is java.awt.Color`
In addition there are defined many enumerations as data types for properties as well as for parameters of methods:

\texttt{\textit{e.g. property PenStyle [integer] PenStyle}}

declares a property which has the enumeration type "PenStyle" and which is named "PenStyle". Data types in square-brackets, like "[integer]" in the example above, are not of interest for Java users. They describe the data type for the ActiveX and VCL controls.

Example on using the above property: \texttt{Doc.PenStyle = PenStyle.Solid}

i.e. the property named \textit{PenStyle} is assigned the value \textit{Solid} of a pre-defined enumeration, which has the type \textit{PenStyle}.

\textbf{Methods:}

The keyword \textit{method} declares a method, \textit{e.g. method void CloseProgressBar()}. Or \texttt{method boolean WriteDoc(string FileName)}

Please note that for methods, according to the Java naming conventions, the first letter is always lower-case. So for Java, \texttt{void CloseProgressBar()} translates into \texttt{void closeProgressBar()}.

\textbf{Properties:}

The keyword \textit{property} describes a property. In many programming languages a property is accessed like a class member variable, \textit{e.g. you can write Vpe.AutoDelete = true. But when this code is executed, in fact a setter-method is called behind the scenes, i.e. Vpe.setAutoDelete(true)}. The same applies to reading the value of a property, behind the scenes a getter-method is called, \textit{e.g. for code like status = Vpe.AutoDelete the following getter is called: status = Vpe.getAutoDelete()}. Unfortunately, Java does not provide this elegant coding technique. Instead of properties, it does only provide getter- and setter-methods.

So when you encounter the keyword \textit{property} in this manual, you need to translate it into getter- and setter-methods. The line below the declaration of the property specifies, how the property can be accessed. Each property is marked for read only (only a getter is present), write only (only a setter is present) or read / write (a getter and a setter is present).

It is also indicated, if the property can be accessed during runtime only, or also during design time. “Design Time” means: when the document is closed and “runtime” means: when the document is open, \textit{i.e. the method Vpe.openDoc() has been called}. The term “Design Time” relates to programming tools with visual designers, where properties of graphical components can be modified by point-and-click during the design time of an application, for example with a Property Grid.
Example:

property boolean PageScrollerTracking
read / write; design- & runtime
means: the property is of type boolean, its name is PageScrollerTracking and it can be accessed for read (a getter is present) and for write (a setter is present) and it is available during runtime (when the document is open) and design time (when the document is closed).

For Java, this translates into the following getter and setter methods:

boolean getPageScrollerTracking ()
void setPageScrollerTracking (boolean value)

If the first letter of a property is lower-case, it must be converted to upper-case for getters and setters.

property VpeCoord nLeft
Translates into the following getter and setter methods:

double getNLeft ()
void setNLeft (double value)

The following members of the class VpeControl can be used as coordinates, if they are used read-only:

public static final double nFree = -2147483549;
pubilc static final double nLeft = -2147483550;
pubilc static final double nRight = -2147483551;
pubilc static final double nLeftMargin = -2147483552;
pubilc static final double nRightMargin = -2147483553;
pubilc static final double nTop = -2147483554;
pubilc static final double nBottom = -2147483555;
pubilc static final double nTopMargin = -2147483556;
pubilc static final double nBottomMargin = -2147483557;

Example:

doc.picture(1, 1, doc.nFree, doc.nFree, “sample.png”);
1.4 VPE ActiveX

The common sequence of function calls is:

- Set the properties for behavior and appearance of VPE in the design mode or during runtime before calling the method OpenDoc
- Open a document with the method "OpenDoc"
- Use all possible methods to insert VPE objects (like Write(), etc.)
- Use "PageBreak" to generate new pages
- Use "Preview" to show the preview to the user, this is optional
- Use "PrintDoc" to print the document, or WriteDoc() to export it
- Close the document with "CloseDoc"

It is only possible to open one document per VPE-ActiveX. If you want to open multiple documents simultaneously, you need to place the same number of VPE-ActiveX's on the form.

Example for Visual Basic, which can be easily translated to other programming languages:
The following example assumes that you use a VPE-ActiveX object named "Report".

```vbnet
Private Sub Form_Load()
    Report.OpenDoc
    Report.VpePrint 1, 1, "Hello World!"
    Report.Preview
End Sub
```

This example is very simple. The document is opened, the text "Hello World!" is inserted at position (1, 1) and afterwards the preview is shown.

Informational: the VPE ActiveX is using the Apartment Threading Model.

For a detailed explanation of the basic programming techniques, please continue reading in the "Programmer's Manual" chapter 3 "Programming Techniques".
1.4.1 Where are the constants like 'VORIENT PORTRAIT' defined?

The special ActiveX / VCL constants are automatically defined, when using the ActiveX or VCL. Both export these constants to your source code.

But some Containers (like for example Visual FoxPro) do not import the constants (like VORIENT PORTRAIT, VFREE, ALIGN_LEFT, etc.).

For a few selected Containers we included such definition files for import. Please check the source code directories.

1.4.2 Important Note for Users of Visual Basic

Visual Basic has problems with the keywords "Print, Write, Line and Scale". VB doesn't recognize that these are methods and properties of an ActiveX. So we implemented them twice. In VB use "VpePrint, VpeWrite, VpeLine and VpeScale" instead.

1.4.3 Understanding the Data Types and Declarations (Visual Basic)

Because the properties and methods of the VPE Control (for .NET, Java, PHP, Python, Ruby, ActiveX and Delphi / C++ Builder VCL) are nearly 100% the same, we explain the API of the VPE control for all the different programming languages within this single book. We use a meta-description to describe the data types, properties, methods and parameters of the VPE API.

Here is how the meta-description is converted to Visual Basic:

Throughout this manual, methods and parameters are always declared in the form <Data Type> <Name>, e.g. boolean AutoDelete would be declared in VB as Dim AutoDelete as Boolean.

The data type in square-brackets defines the data type for the ActiveX.

Example:

property PenStyle [integer] PenStyle

means that this property is of type "integer".

The leading "PenStyle" type declaration is for the .NET control only.

Data Types:

VpeCoord is Double
boolean is Boolean
integer is Integer
long is Long
string is String
Color is Color (in fact a 32-bit long integer)
Properties / Methods:
The keyword `method` declares a method, e.g. `method void CloseProgressBar()`. 

A method is either a Sub or a Function, depending on how the return type of the method is declared. The keyword immediately following the word `method` declares the method.

`void` is a `Sub` which does not return a value, e.g. 
`method void StorePos()` is in Visual Basic: `Sub StorePos()`

all other declarations declare a function, e.g. 
`method boolean WriteDoc(string FileName)` is in Visual Basic 
`Function WriteDoc(FileName as String) as Boolean`

The keyword `property` describes a property. The line below the declaration of the property explains, how the property can be accessed (read-only, write-only or read / write together) and if it can be accessed during runtime only, or also during design time.

Example:

`property boolean PageScrollerTracking` 
`read / write; design- & runtime`

means: the property is of type `Boolean`, its name is `PageScrollerTracking` and it can be accessed for `read` (you can read its value, e.g. `n = PageScrollerTracking`) and for `write` (you can assign it a value, e.g. `PageScrollerTracking = True`) and it is available during runtime (while your application is running) and design time (when you are developing your forms).

It would be declared in Visual Basic as `Dim PageScrollerTracking as Boolean`. 
1.4.4 Exceptions

In case of error conditions, the VPE ActiveX might raise errors. We call that "throw exceptions". Properties and methods of the VPE ActiveX which raise errors are marked in this reference.

Visual Basic for example offers three statements to handle these errors:

- On Error GoTo line
- On Error Resume Next
- On Error GoTo 0

Visual Basic: please note that `Err.Number` needs to be compared with the VPE Exception Code + the constant value `vbOBJECTERROR`.

```
Select Case Err.Number
    case VPE_E_APIFAILED + vbOBJECTERROR 'Error caused by VPE API
        If VPE.LastError = VERR_TPL_AUTHENTICATION Then
            MsgBox("'sample2.dcd' has been modified.", "Error:", MB_OK);
        ElseIf VPE.LastError = VERR_FILE_OPEN Then
            MsgBox("'sample2.dcd' not found or no access rights.", "Error:", MB_OK);
        Else
            MsgBox("'sample2.dcd' could not be read.", "Error:", MB_OK);
        End If
    case Else
        'Handle other situations here...
End Select
End Sub
```

Please consult the documentation of your programming language / tool for further details on how to handle errors.

Possible VPE Exception Codes raised by the ActiveX are:

<table>
<thead>
<tr>
<th>Exception Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPE_E_NEEDOPENDOCUMENT</td>
<td>1024 The VPE Document is not open. You need to call OpenDoc prior to this operation.</td>
</tr>
<tr>
<td>VPE_E_NEEDCLOSEDDOCUMENT</td>
<td>1025 The VPE Document is open. The document needs to be closed prior to this operation.</td>
</tr>
<tr>
<td>VPE_E_ENGINEAPI</td>
<td>1026 An underlaying API routine was not found in the VPE dynamic link library.</td>
</tr>
<tr>
<td>VPE_E_OUTOFRANGE</td>
<td>1027 The argument is out of range.</td>
</tr>
<tr>
<td>VPE_E_NOTIMPLEMENTED</td>
<td>1028 Sorry, this feature is not implemented in this edition of VPE.</td>
</tr>
<tr>
<td>VPE_E_NOWINDOW</td>
<td>1029 Need a window to perform this operation. The object must have been in-place activated first.</td>
</tr>
<tr>
<td>VPE_E_APIFAILED</td>
<td>1030 An underlaying VPE API routine has failed. Possible causes: Invalid arguments, conversion of data type impossible, Out of Memory, etc.</td>
</tr>
<tr>
<td>VPE_E_INVALIDOBJECTHANDLE</td>
<td>1031 The handle to the object is invalid: it does not point to a valid VPE DLL object. Probably the method/property that returned the object has failed.</td>
</tr>
</tbody>
</table>
VPE_E_VPEGETFAILED 1032
The value could not be retrieved. Possible causes: Field/Control not found, Data Type Conversion Impossible or Out of Memory.

VPE_E_VPESETFailed 1033
The value could not be set. Possible causes: Field/Control not found, Data Type Conversion Impossible or Out of Memory.

1.4.5 Live Demo on our Websites
On our websites at http://www.IdealSoftware.com/activex/bin/vpectrl.htm (mirrored at http://www.IdealSoftware.de/activex/bin/vpectrl.htm) we implemented a live demo, which demonstrates our VPE ActiveX running in a real internet environment, displaying a VPE document file which is uploaded together with the VPE ActiveX from the server to the client's browser.

You need the Internet Explorer in order to view the demo.
1.4.6 CreateObject Call

The ActiveX supports the CreateObject() call, i.e. it can be called from non-visual classes and objects. This is very useful if running in Active Server Pages (ASP).

Example for Visual Basic:

```vba
Dim x As Object
Set x = CreateObject("VpeControl.VpeControl.72")
```

The important parameter is “VpeControl.VpeControl.72” which is the class name of the VPE ActiveX Control.

1.4.7 Early Binding

The VPE ActiveX can be used with early binding without using a form, which has some performance advantages, for example for batch-processes on a server.

Example for Visual Basic:

**IMPORTANT:** do not insert VPE into your VB Project as Component, but as Reference!

```vba
Dim report As VPE
Private Sub Command1_Click()
    Set report = New VPE
    report.OpenDoc
    report.Preview
End Sub
```

1.4.8 Visual FoxPro Note

To make Visual FoxPro process all VPE events, and to prevent getting sporadic "OLE Exception Error" messages, you need to set in VFP the system variable "_VFP.AutoYield = .F."

If you want that a form which contains the VPE ActiveX is closed when the Close Button in VPE's toolbar is clicked, do not call `thisform.release` when processing the event `AfterDestroyWindow()`, because VFOX will GPF for an unknown reason. Workaround: hide the Close Button in VPE's toolbar by setting `tbClose = .F.` - since the VFP form already has a Close Button.

Otherwise, there is the following workaround: when receiving the `AfterDestroyWindow()` event fired by VPE, start a timer with 0.1 second delay. After the timer has been counted down, you may call `thisform.release` safely.
1.4.9 MFC: Using the VPE ActiveX with the MFC without placing it in a Dialog Resource

If you didn’t place the VPE ActiveX as Resource into a Dialog, you need to implement the COM licensing mechanism of Microsoft by yourself (see the according documentation from Microsoft).

**In short:**

Call `CWnd::CreateControl()` - in the last parameter you must specify the license key **generated** by the control. In order to get the **generated** key, you once need to create during development a project with the VPE ActiveX and call the methods:

```c
lpU = CWnd::GetControlUnknown()
lpU -> QueryInterface(IID_IClassFactory2, &pClassFactory2)
pClassFactory2->RequestLicKey(NULL, pBstrKey)
```

`pBstrKey` will contain the generated key, which is the coded License Key. Use this key in all other calls to `CWnd::CreateControl()` in your projects.

If you place the VPE ActiveX in VC++ as resource in a dialog you will have no problems and you don’t need to get the key, etc. because VC does everything for you.#

1.4.10 Using the VPE ActiveX in a CFormView derived class

Q: If I use the VPE ActiveX in a CFormView derived class, the control is not updated after a call to `CFormView::Invalidate()` or `CDocument::UpdateAllViews()`.

A: Set the style of the dialog resource containing the VPE ActiveX to "Clip Children" (do not change the style in `PreCreateWindow()`). You can also call `CVpeControl::UpdateWindow()` in the overridden `CFormView::OnUpdate()` method.
1.5 VPE VCL

The common sequence of function calls is:

- Set the properties for behavior and appearance of VPE in the design mode, or during runtime before calling OpenDoc
- Open a virtual document with the method "OpenDoc"
- Use all possible methods to insert VPE objects (like Write(), etc.)
- Use "PageBreak" to generate new pages
- Use "Preview" to show the preview to the user, this is optional
- Use "PrintDoc" to print the document, or WriteDoc() to export it
- Close the document with "CloseDoc"

It is only possible to open one document per VPE-VCL. If you want to open multiple documents simultaneously, you need to place the same number of VPE-VCL’s on the form.

The following example assumes that you use a VPE-VCL object named "Report".

```pascal
procedure TForm1.FormCreate(Sender: TObject);
begin
    Report.OpenDoc;
    Report.Print(1, 1, 'Hello World!');
    Report.Preview;
end;
```

This example is very simple. The document is opened, the text "Hello World!" is inserted at position (1, 1) and afterwards the preview is shown.

For a detailed explanation of the basic programming techniques, please continue reading in the "Programmer's Manual" chapter 3 "Programming Techniques".

If you are using the VPE Enterprise or Interactive Edition, there is also a very important note in the Programmer's Manual: Important Note for VPE-VCL Users!
1.5.1 Understanding the Data Types and Declarations (Delphi)

Because the properties and methods of the VPE Control (for .NET, Java, PHP, Python, Ruby, ActiveX and Delphi / C++ Builder VCL) are nearly 100% the same, we explain the API of the VPE control for all the different programming languages within this single book. We use a meta-description to describe the data types, properties, methods and parameters of the VPE API.

Here is how the meta-description is converted to Delphi:

Throughout this manual, methods and parameters are always declared in the form <Data Type> <Name>, e.g. **boolean AutoDelete** would be declared in Delphi as **AutoDelete: Boolean**.

The data type in square-brackets defines the data type for the VCL Component.

**Example:**

```
property PenStyle [integer] PenStyle
```

means that this property is of type "integer".

The leading "PenStyle" type declaration is for the .NET control only.

**Data Types:**

- **VpeCoord is Double**
- **boolean is Boolean**
- **integer is Integer**
- **pointer is Pointer**
- **long is Long**
- **string is String**
- **Color is TColor** (in fact a 32-bit integer)

**Properties / Methods:**

The keyword **method** declares a method, e.g. **method void CloseProgressBar()**.

A method is either a Procedure or a Function, depending on how the return type of the method is declared. The keyword immediately following the word **method** declares the method.

- **void is a Procedure** which does not return a value, e.g. **method void StorePos()** is in Delphi: **procedure StorePos()**
- all other declarations declare a function, e.g. **method boolean WriteDoc(string FileName)** is in Delphi **function WriteDoc(FileName: String) : Boolean**

The keyword **property** describes a property. The line below the declaration of the property explains, how the property can be accessed (read-only, write-only or read / write together) and if it can be accessed during runtime only, or also during design time.
Example:

```delphi
property boolean PageScrollerTracking
read / write; design- & runtime
```

means: the property is of type `Boolean`, its name is `PageScrollerTracking` and it can be accessed for `read` (you can read its value, e.g. `n := PageScrollerTracking`) and for `write` (you can assign it a value, e.g. `PageScrollerTracking := True`) and it is available during runtime (while your application is running) and design time (when you are developing your forms).

It would be declared in Delphi as `PageScrollerTracking: Boolean`

### 1.5.2 Exceptions

In case of error conditions, the VPE-VCL might raise exceptions. We call that "throw exceptions". Properties and methods of the VPE-VCL which throw exceptions are marked in this reference. Please consult the Object Pascal and VCL documentation for further details on how to handle exceptions.

In order to throw and handle exceptions, the class `EVPEError` is defined by the VPE-VCL:

```delphi
TVPEVCL_ERR =
(
    VPEVCL_ERR_DOC_NOT_OPEN,
    VPEVCL_ERR_DOC_IS_OPEN,
    VPEVCL_ERR_GET_VALUE,
    VPEVCL_ERR_SET_VALUE,
    VPEVCL_ERR_TPL_LOAD,
    VPEVCL_ERR_NULL_HANDLE
);
EVPEError = class(Exception)
protected
    ErrCode: TVPEVCL_ERR;
public
    constructor CreateByCode(code:TVPEVCL_ERR);
    property ErrorCode: TVPEVCL_ERR read ErrCode;
end;
```

If an exception of type `EVPEError` is raised, you can catch it and retrieve the thrown exception code with the property `EVPEError.ErrCode`.

**Possible values for `ErrCode` are:**

- **VPEVCL_ERR_DOC_NOT_OPEN**
  The VPE Document is not open. You need to call `OpenDoc` prior to this operation.

- **VPEVCL_ERR_DOC_IS_OPEN**
  The VPE Document is open. The document needs to be closed prior to this operation.

- **VPEVCL_ERR_GET_VALUE**
  The value could not be retrieved. Possible causes: Field/Control not found, Data Type Conversion Impossible or Out of Memory.
- **VPEVCL_ERR_SET_VALUE**
  The value could not be set. Possible causes: Field/Control not found, Data Type Conversion Impossible or Out of Memory.

- **VPEVCL_ERR_TPL_LOAD**
  The template file could not be loaded. Possible causes: file not found, wrong version or VPE Edition, or Out of Memory.

- **VPEVCL_ERR_NULL_HANDLE**
  The handle to the object is invalid. It does not point to a valid VPE DLL Object. Probably the method/property that returned the object has failed.

After catching the Exception Code, you can check for further error information with the property `VPE.LastError`. 
2 Events Generated by the VPE - .NET Control

2.1 Events Generated by VPE - .NET Control

The .NET Control fires several events to your application, so you have always total control about what's happening.

NOTE: You may not call `CloseDoc()` while processing any event fired by VPE - except it is explicitly said in the description of a particular event that it is allowed.
2.2 AfterDestroyWindow Event - .NET

[Not supported by VpeWebControl]

Is fired when the preview window was destroyed - for example closed by the user - and AutoDelete is True (the default). The document is also closed (removed from memory).

**EventHandler VPE.AfterDestroyWindow**(
   object sender,
   System.EventArgs e
)

*object sender*
   the VPE document object that fired the event

*System.EventArgs e*
   event data, see .NET documentation

**Remarks:**

Do not call any VPE method or property when processing this event. The document is already closed and not accessible.

This event is fired, if AutoDelete is True and the user closes the preview. Otherwise the event AfterCloseWindow is fired.
2.3 Closing Event - .NET

[Not supported by VpeWebControl]

VPE requests confirmation from your application, if the preview can be closed.

CancelButtonEventHandler VPE.Closing(
    object sender,
    System.ComponentModel.CancelEventArgs e
)

*object sender*
  the VPE document object that fired the event

*System.ComponentModel.CancelEventArgs e*
  event data, see .NET documentation

You can set `e.Cancel = true` if you want to cancel the event, i.e. deny that the preview is closed

Example:

```vbnet
Protected Sub vpe_Closing(sender As Object, e As CancelEventArgs)
    If Not myDataIsSaved Then
        e.Cancel = True
        MessageBox.Show("You must save first.")
    Else
        e.Cancel = False
        MessageBox.Show("Goodbye.")
    End If
End Sub 'vpe_Closing
```
2.4 AfterCloseWindow Event - .NET

[Not supported by VpeWebControl]

Is fired, when the preview window was closed - for example by the user - and AutoDelete is False (which means, that the document is not destroyed, if the preview is closed).

Event

EventHandler VPE. AfterCloseWindow(
    object sender,
    System.EventArgs e
)

object sender
    the VPE document object that fired the event

System.EventArgs e
    event data, see .NET documentation

See also:
    AfterDestroyWindow Event - .NET
2.5 BeforeOpenFile Event - .NET

[Not supported by VpeWebControl]

Is fired when the user clicked the Open File button in the toolbar (or pushed the corresponding key).

```csharp
object sender
System.ComponentModel.CancelEventArgs e

Protected Sub vpe_BeforeOpenFile (sender As Object, e As CancelEventArgs)
    e.Cancel = True
End Sub
```

**See also:**

OpenFileName
2.6 AfterOpenFile Event - .NET

[Not supported by VpeWebControl]

Is fired after the file open operation has been completed.

EventHandler VPE. AfterOpenFile(  
    object sender,  
    System.EventArgs e  
)

object sender  
    the VPE document object that fired the event

System.EventArgs e  
    event data, see .NET documentation

Remarks:
When handling this event, your application can acquire the error status of the file open operation by reading the property LastError. For example, if the user clicked onto the Cancel-Button in the open file dialog, LastError will be ErrorCode.Cancelled.

See also:
OpenFileName
2.7 BeforeSaveFile Event - .NET

[Not supported by VpeWebControl]

Is fired when the user clicked the Save File button in the toolbar (or pushed the corresponding key).

**CancelEventHandler** VPE. BeforeSaveFile(
    object sender,
    System.ComponentModel.CancelEventArgs e
)

*object sender*
the VPE document object that fired the event

*System.ComponentModel.CancelEventArgs e*
event data, see .NET documentation

You can set `e.Cancel = true` if you want to cancel the event, i.e. deny that the file save dialog will be shown. Cancelling the operation allows you to display your own save dialog and to export your own document, for example to a memory stream and from there to a database.

**Example:**

```vbnet
Protected Sub vpe_BeforeSaveFile (sender As Object, e As CancelEventArgs)
    e.Cancel = True
End Sub
```

**See also:**

*SaveFileName*
2.8 AfterSaveFile Event - .NET

[Not supported by VpeWebControl]

Is fired after the file save operation has been completed.

EventHandler VPE. AfterSaveFile(
    object sender,
    System.EventArgs e
)

object sender
    the VPE document object that fired the event

System.EventArgs e
    event data, see .NET documentation

Remarks:
When handling this event, your application can acquire the error status of the file open
operation by reading the property LastError. For example, if the user clicked onto the Cancel-
Button in the save file dialog, LastError will be ErrorCode.Cancelled.

See also:
SaveFileName
2.9 HelpRequested Event - .NET

[Not supported by VpeWebControl]

Is sent if the property `RouteHelp` is True and the user clicked the Help-Button in the toolbar or pushed the corresponding key (F1 by default).

This is one of the few events where `CloseDoc()` may be called while processing it.

```csharp
HelpEventHandler VPE.HelpRequested(
    object sender,
)
```

- `object sender` 
  - the VPE document object that fired the event
  - event data, see .NET documentation
2.10 AfterAutoPageBreak Event - .NET

Is sent, when a Auto Page Break occurred (see "Automatic Text Break" in the Programmer's Manual). The engine is on the new page. This is either a newly generated page, or an already existing page (if the text was inserted on a page, that has already pages following).

This gives you additional control over the layout. When the event is fired, VPE already moved to the next page (or generated one) and you can modify the Output Rectangle (NOT the Default Output Rectangle) to control the layout, or insert manually Headers and Footers, etc. (see "Headers and Footers" in the Programmer's Manual).

```csharp
EventHandler VPE.AfterAutoPageBreak(
    object sender,
    System.EventArgs e
)
```

- **object sender**
  - the VPE document object that fired the event

- **System.EventArgs e**
  - event data, see .NET documentation

**Remarks:**
If your application is not able to receive events, or if you don't want to process this event for some reason, you can test for AutoBreak (this means, check if an AutoBreak occurred after inserting a text object) with the following technique:

Store the current page-number in a variable. After the output, compare this variable to the current page-number. If it is different, an AutoBreak had occurred.

**Code example:**

```csharp
n = Doc.CurrentPage // remember the current page number
Doc.Print(1, 1, "... very long text ...")
if n <> Doc.CurrentPage
    caught auto break!
    The formula "Doc.CurrentPage - n" returns the number of automatically generated pages.
```

**Notes:**
If you are inserting text objects into the document while processing this event, make sure they will not cause again an Auto Page Break event - otherwise this will cause an endless recursion, ending up in a stack overflow with a GPF.

**To avoid Auto Page Breaks, set AutoBreakMode = AUTO_BREAK_NO_LIMITS.**

If you are modifying any properties, as for example the AutoBreakMode or FontSize etc., it will not be reset when you code exits the event handler. If you are changing properties, you need to save and restore them by code yourself. Do this by calling StoreSet() in the beginning of you event handler and by calling UseSet() followed by RemoveSet() at the end.

Example: if a Print() statement in your main code causes an AutoBreak and if your AutoBreak Handler sets the PenSize to something different than zero, then the rest of the text which is output by the previous Print() command of your main code will have a box around it.

Solution: at the very first beginning of your AutoBreak Handler call StoreSet() and on the last line call UseSet() and then RemoveSet(). Do this only if required, you can also check which single properties have been changed and change them back, because StoreSet() and UseSet() eat performance!
YOU MAY NOT USE NoHold of the embedded flags in the AutoBreak-Event Handler. This will cause an internal recursion and overwrite settings done in the main code. Explanation: When using the flag NoHold in your main code in a Print(Box)() or Write(Box)() statement which causes an AutoBreak, the current settings are stored in a temporary memory block (to be restored after the command has been executed). Afterwards your AutoBreak Event-Handler is executed and if you use the NoHold Flag again, the temporary memory block is overwritten, so that its initial settings are lost.
2.11 RequestPrint Event - .NET

Is fired to inform the application about the several stages during the printing process.

```csharp
RequestPrintEventHandler VPE.RequestPrint(
    object sender,
    RequestPrintEventArgs e
)
```

- **object sender**
  - the VPE document object that fired the event

- **RequestPrintEventArgs e**
  - event data, derived from EventArgs, two additional members:
    ```csharp
    public class RequestPrintEventArgs : EventArgs
    {
        public PrintAction Action;
        public PrintResultingAction PrintResultingAction;
    }
    ```

**PrintAction enumeration:**
```csharp
public enum PrintAction
{
    Abort,  // User aborted while printing
    Start,  // Print started
    End,    // Print ended
    SetupAbort,  // User aborted Setup-Dialog
    SetupStart,  // Setup-Dialog started
    SetupEnd,   // Setup-Dialog ended
}
```

**PrintResultingAction enumeration:**
is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:
```csharp
public enum PrintResultingAction
{
    Ok,
    Abort,
}
```

**Remarks:**
Do not call `CloseDoc()` while processing this event. You would terminate a module that is working.

When your application handles this event, the `PrintAction` member contains the current status of the printing progress. Your application’s event handler can return a value in the `PrintResultingAction` member.

Your application should return `Ok` if it processes this message, except for `PrintAction = SetupStart`, where your application may return in addition `Abort` to abort the print job. `SetupStart` is sent, when the user clicked the print button in the preview (or pushed the corresponding key). You have the option to abort the job, for you can then create and print internally a new document which is completely different to the preview.
Another use for the SetupStart message is, to pre-initialize the Device Control Properties at this stage, before the printer setup dialog will be shown to the user.
2.12 BeforePrintNewPage Event - .NET

Is fired only while printing exactly before printing a new page. The event is useful to change the Device Control Properties on-the-fly during printing. You may change all properties, except the device itself.

```csharp
PrintEventHandler VPE.BeforePrintNewPage(
    object sender,
    PrintEventArgs e
)
```

- `object sender` - the VPE document object that fired the event
- `PrintEventArgs e` - event data, derived from EventArgs, two additional members:
  ```csharp
  public class PrintEventArgs : EventArgs
  {
      public int Page;
      public PrintResultingAction PrintResultingAction;
  }
  ```
  - `int Page` - current page number that will be printed
  - `PrintResultingAction` enumeration:
    - is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:
    ```csharp
    public enum PrintResultingAction
    {
        Ok,
        Change,
    }
    ```

**Remarks:**
Your application’s event handler can return a value in the ResultingAction member of the PrintEventArgs parameter.

Your application should return Ok, if it processes this message without changing a printing device’s properties.

If a Device Control Property was changed, your application must return Change.

- If you change the Device Control Properties during the print job, you must reset them to the original values, when the print job has finished (see RequestPrint Event – .NET: with PrintAction = PrintAction.Abort or PrintAction = PrintAction.End).
- Changing the properties (like DevPaperBin, DevOrientation, DevPaperFormat, etc.) during the print job doesn’t work with some (buggy) printer drivers, for example some fax drivers and the HP4 M Plus driver (the HP4 PS driver should work with the HP4 M Plus printer!).
- Some properties and methods don’t work with some printer drivers. For example "DevTTOption" doesn’t work with our HP4 and HP5 printer drivers on WfW 3.11 and NT 3.51, but it works with both drivers on Win95. This is a driver problem.
Known Problems:

Changing any Device Control Properties during the print job disables Duplex Printing on PCL Printers

SYMPTOMS
When you change a Device Control Property during the print job, it appears to disable Duplex (double-sided) printing when the target printer is a (Hewlett Packard) PCL printer.

CAUSE
PCL printers treat a change in paper size as a new print job that requires the printer to be initialized. This causes the printer to eject any page that is currently in the printer. The PCL printer drivers for Windows assume that the page size has been changed when a Device Control Property is changed during the print job, unless the orientation of the page has changed.

RESOLUTION
To prevent having a page ejected when changing a Device Control Property during the print job, make sure that the function is called only between individual sheets of paper. Changing a Device Control Property before printing odd-numbered pages is sufficient for most applications that use duplex printing. However, some applications require that you change the page orientation on a page-by-page basis. In this case, you can change Device Control Properties between individual sheets of paper if the orientation has changed.

STATUS
This behavior is by design.

MORE INFORMATION
Note that when this problem occurs the print job continues and the sheets of paper are passed through the printer's duplexer, but the sheets are only printed on one side.

Because of the page size initialization requirement for PCL printers, Windows PCL drivers treat the change of Device Control Properties differently. These drivers allow only the orientation to change between the front and back pages of a sheet of paper. This means that the change of a Device Control Property will eject the page unless the orientation (and only the orientation) has changed from the previous page. Returning PrintResultingAction.Change although no Device Control Property has been changed causes the printer to eject the page.

Returning PrintResultingAction.Change although no Device Control Property has been changed is unnecessary. By doing so, you risk having a page ejected from the printer, which has a high probability of occurring.

See also:
PrintDevData Event - .NET
2.13 PrintDevData Event - .NET

Is sent only while printing, exactly before printing a new page and immediately after BeforePrintNewPage() has been sent. The only use for this event is to call DevSendData() in response.

DevSendData() enables your application to send escape sequences to the printing device. So it is possible to select for example an output paper bin by code (or whatever other functionality is provided by the connected printer).

**PrintEventHandler VPE.PrintDevData**

```csharp
object sender, 
PrintEventArgs e
)
```

*object sender*
  the VPE document object that fired the event

*PrintEventArgs e*
  event data, derived from EventArgs, two additional members:
  ```csharp
  public class PrintEventArgs : EventArgs
  {
    public int Page;
    public PrintResultingAction PrintResultingAction;
  }
  ```

*int Page*
  current page number that will be printed

*PrintResultingAction enumeration:*
  is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:
  ```csharp
  public enum PrintResultingAction
  {
    Ok,
    Change,
  }
  ```

**Remarks:***
  When your application handles this event, the PrintAction member contains the current status of the printing progress. Your application’s event handler can return a value in the PrintResultingAction member.

  Your application should return **Ok** if it processes this message without calling DevSendData().

  If it called DevSendData(), your application must return **Change**.

**See also:***
  BeforePrintNewPage Event
2.14 BeforeMail Event - .NET

[Not supported by VpeWebControl]

Is fired, when the user clicked the eMail-button in the preview (or pushed the corresponding key). The e-mail was not sent yet, therefore your application has now the option to set receivers, attachments, etc. by code.

CancelEventHandler VPE. BeforeMail(
    object sender,
    System.ComponentModel.CancelEventArgs e
)

object sender
    the VPE document object that fired the event
System.ComponentModel.CancelEventArgs e
    event data, see .NET documentation

You can set e.Cancel = true if you want to cancel the event, i.e. deny that the document will be mailed. Cancelling the operation allows you to display your own mail dialog and/or to execute your own mailing code.

Example:

```
Protected Sub vpe_BeforeMail (sender As Object, e As CancelEventArgs)
    e.Cancel = True
    // Here comes your own code to mail a document
End Sub
```

See also:

E-Mail Functions
2.15 AfterMail Event - .NET

[Not supported by VpeWebControl]

Is sent, after the user clicked the eMail-button in the preview (or pushed the corresponding key) and the e-mail has already been sent.

```csharp
EventHandler VPE.AfterMail (  
    object sender,
    System.EventArgs e
)
```

- **object sender**
  - the VPE document object that fired the event

- **System.EventArgs e**
  - event data, see .NET documentation

**Remarks:**
When handling this event, your application can acquire the error status of the mail by reading the property `LastError`.

**See also:**
E-Mail Functions
2.16 ObjectClicked Event - .NET

[Not supported by VpeWebControl, Professional Edition and above]

A clickable object with an assigned ObjectID has been clicked with the mouse.
This is one of the few events where CloseDoc() may be called while processing the event.

ObjectClickedEventHandler VPE.ObjectClicked(
    object sender,
    ObjectClickedEventArgs e
)

object sender
    the VPE document object that fired the event

ObjectClickedEventArgs e
    event data, derived from EventArgs, one additional member:
        public class ObjectClickedEventArgs : EventArgs
        {
            public int ObjectID;  // Object ID of the object that was clicked by the user
        }

See also:
    Clickable objects


2.17 UDOPaint Event - .NET

[Professional Edition and above]

Is fired as a notification from a User Defined Object (UDO). The object needs to be painted to the output device.

See the description of the User Defined Objects for information on how to process this event.

```csharp
EventHandler VPE.UDOPaint(
    object sender,
    System.EventArgs e
)
```

- `object sender` - the VPE document object that fired the event
- `System.EventArgs e` - event data, see .NET documentation

See also:

- User Defined Object
2.18 AfterControlEnter - .NET

[Not supported by VpeWebControl, Interactive Edition and above]

A Control received the focus.
Normally, this event is sent if the user presses the Tab or Backtab key, or clicks with the mouse onto a control. This event can be used to re-format the content of a control.

```csharp
VPEObjectEventHandler VPE.AfterControlEnter(
    object sender,
    VPEObjectEventArgs e
)
```

*object sender*

the VPE document object that fired the event

*VPEObjectEventArgs e*

event data, derived from EventArgs, one additional member:

```csharp
public class VPEObjectEventArgs : EventArgs
{
    public TVPEObject VpeObject; // the VPE object that fired the event
}
```

See also:

Interactive Documents
2.19 RequestControlExit - .NET

[Not supported by VpeWebControl, Interactive Edition and above]

The user wishes to remove the focus from the currently focused Control. In response to this event your application can evaluate the value of the Control and decide, whether the current value is valid and the Control may lose the focus or not.

**ControlExitEventHandler** VPE.RequestControlExit(

  object sender,
  ControlExitEventArgs e
)

*object sender*
  
  the VPE document object, which fired the event

*ControlExitEventArgs e*
  
  event data, derived from EventArgs, two additional members:

  ```
  public class ControlExitEventArgs : EventArgs
  {
    public TVPEObject VpeObject; // the VPE object that fired the event
    public bool CanExit;
  }
  ```

*bool CanExit*
  
  is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, the control may loose the focus</td>
</tr>
<tr>
<td>False</td>
<td>no, the control may not loose the focus</td>
</tr>
</tbody>
</table>

**See also:**

[Interactive Documents](#)
2.20 AfterControlExit - .NET

[Not supported by VpeWebControl, Interactive Edition and above]

The currently focused Control lost the focus.
Normally, this event is sent if the user presses the Tab or Backtab key, or clicks with the mouse onto another control. When receiving this message, it is possible for you to force the focus to be set explicitly to a specific control by calling SetFocusControlByName() or SetFocus(). It is also possible to enable and disable other controls of the current form while processing this event.
In addition this event can be used to re-format the content of a control.

VPEObjectEventHandler VPE.AfterControlExit(
    object sender,
    VPEObjectEventArgs e
)

object sender
    the VPE document object that fired the event

VPEObjectEventArgs e
    event data, derived from EventArgs, one additional member:
    public class VPEObjectEventArgs : EventArgs
    {
        public TVPEObject VpeObject; // the VPE object that fired the event
    }

See also:
    Interactive Documents
2.21 AfterControlChange - .NET

[Not supported by VpeWebControl, Interactive Edition and above]

A Control changed its value, i.e. the content of a Control was edited by the user or the value of an associated Field was changed by code.

Evaluating this event means, that your application is informed about every single keystroke or mouse-click, which modifies a Control's content.

If you are working with Fields that are associated with controls - as recommended - your application should not take care of this event.

This is one of the few events where CloseDoc() may be called while processing the event.

The event is not fired, if you set the value of a Control by code.

```
VPEObjectEventHandler VPE.AfterControlChange(
    object sender,
    VPEObjectEventArgs e
)
```

*object sender*
- the VPE document object that fired the event

*VPEObjectEventArgs e*
- event data, derived from EventArgs, one additional member:
  ```
  public class VPEObjectEventArgs : EventArgs
  {
      public TVPEObject VpeObject; // the VPE object that fired the event
  }
  ```

Remarks:
If you change the value of a Control by code, the AfterControlChange event is not fired. But if the Control is associated with a Field, the event AfterFieldChange is fired.

Vice versa, if you change the value of a Field by code, the AfterFieldChange event is not fired, but any Controls associated with the Field will fire the event AfterControlChange.

See also:
  Interactive Documents
2.22 AfterFieldChange - .NET

[Not supported by VpeWebControl, Interactive Edition and above]

A Field (not a control!) changed its value, because the associated Control was edited by the user or the content of any associated Control was changed by code. This event is very interesting, because a Field will change its value each time the user makes a change. Evaluating this event means, that your application is informed about every single keystroke or mouse-click, which modifies a Control's content. This is one of the few events where CloseDoc() may be called while processing the event. The event is not fired, if you set the value of a Field by code.

VPEFieldEventHandler VPE.AfterFieldChange(
    object sender,
    VPEFieldEventArgs e
)

object sender
    the VPE document object that fired the event
VPEFieldEventArgs e
    event data, derived from EventArgs, one additional member:
    public class VPEFieldEventArgs : EventArgs
    {
        public TVPEField Field;        // the Field object that fired the event
    }

Remarks:

If you change the value of a Control by code, the AfterControlChange event is not fired. But if the Control is associated with a Field, the event AfterFieldChange is fired. Vice versa, if you change the value of a Field by code, the AfterFieldChange event is not fired, but any Controls associated with the Field will fire the event AfterControlChange.

See also:

Interactive Documents
3 Events Generated by the VPE - Java Control

3.1 Events Generated by VPE - Java Control

The Java Control fires several events to your application, so you have always total control about what's happening.

**NOTE:** You may not call `CloseDoc()` while processing any event fired by VPE - except it is explicitly said in the description of a particular event that it is allowed.

The event handling follows the Java standard. You need to implement an event listener class, which must be derived from `VpeEventAdapter`.

Depending on the edition of VPE, `VpeEventAdapter` is defined as follows:

```java
// Community, Standard and Enhanced Edition
public class VpeEventAdapter implements VpeEventListener {
    public VpeEventAdapter() {}

    // GUI and Non-GUI
    public void AfterAutoPageBreak(EventObject e) {}

    // GUI only: The following methods are only present in the GUI version
    public void AfterDestroyWindow(EventObject e) {}
    public void AfterCloseWindow(EventObject e) {}
    public void AfterOpenFile(EventObject e) {}
    public void AfterSaveFile(EventObject e) {}
    public void AfterMail(EventObject e) {}
    public void Closing(CancelEvent e) {}
    public void BeforeOpenFile(CancelEvent e) {}
    public void BeforeSaveFile(CancelEvent e) {}
    public void BeforeMail(CancelEvent e) {}
    public void HelpRequested(EventObject e) {}
    public void RequestPrint(RequestPrintEvent e) {}
    public void BeforePrintNewPage(PrintEvent e) {}
    public void PrintDevData(PrintEvent e) {}
}
```
public class VpeEventAdapter implements VpeEventListener {
    public VpeEventAdapter() {} 

    // GUI and Non-GUI
    public void AfterAutoPageBreak(EventObject e) {} 

    // GUI only: The following methods are only present in the GUI version
    public void AfterDestroyWindow(EventObject e) {} 
    public void AfterCloseWindow(EventObject e) {} 
    public void AfterOpenFile(EventObject e) {} 
    public void AfterSaveFile(EventObject e) {} 
    public void AfterMail(EventObject e) {} 
    public void Closing(CancelEvent e) {} 
    public void BeforeOpenFile(CancelEvent e) {} 
    public void BeforeSaveFile(CancelEvent e) {} 
    public void BeforeMail(CancelEvent e) {} 
    public void HelpRequested(EventObject e) {} 
    public void RequestPrint(RequestPrintEvent e) {} 
    public void BeforePrintNewPage(PrintEvent e) {} 
    public void PrintDevData(PrintEvent e) {} 

    // GUI only, Professional Edition
    public void ObjectClicked(ObjectClickedEvent e) {} 

    // Interactive Edition
    public class VpeEventAdapter implements VpeEventListener {
        public VpeEventAdapter() {} 

        // GUI and Non-GUI
        public void AfterAutoPageBreak(EventObject e) {} 

        // GUI only: The following methods are only present in the GUI version
        public void AfterDestroyWindow(EventObject e) {} 
        public void AfterCloseWindow(EventObject e) {} 
        public void AfterOpenFile(EventObject e) {} 
        public void AfterSaveFile(EventObject e) {} 
        public void AfterMail(EventObject e) {} 
        public void Closing(CancelEvent e) {} 
        public void BeforeOpenFile(CancelEvent e) {} 
        public void BeforeSaveFile(CancelEvent e) {} 
        public void BeforeMail(CancelEvent e) {} 
        public void HelpRequested(EventObject e) {} 
        public void RequestPrint(RequestPrintEvent e) {} 
        public void BeforePrintNewPage(PrintEvent e) {} 
        public void PrintDevData(PrintEvent e) {} 

        // GUI only, Professional Edition
        public void ObjectClicked(ObjectClickedEvent e) {} 

        // GUI only, Interactive Edition
        public void AfterControlEnter(VPEObjectEvent e) {} 
        public void AfterControlExit(VPEObjectEvent e) {} 
        public void AfterControlChange(VPEObjectEvent e) {} 
        public void RequestControlExit(ControlExitEvent e) {} 
        public void AfterFieldChange(VPEFieldEvent e) {} 
    }

    Each event is explained in detail throughout the following sections.
The event handling is shown in practice in the Java demo source codes, which are installed together with VPE. Basically, derive a class from `VpeEventAdapter` and overload the methods for which you want to receive events.

Assuming your application is a class named `MyApplication`, your event listener could be:

```java
class MyApplicationListener extends VpeEventAdapter {
    MyApplication adaptee;
    MyApplicationListener(MyApplication adaptee) {
        this.adaptee = adaptee;
    }

    // In this example we are only interested in the AfterDestroyWindow-event
    public void AfterDestroyWindow(EventObject e) {
        super.AfterDestroyWindow(e);
        adaptee.AfterDestroyWindow(e);
    }
}
```

Next, implement for your class `MyApplication` the method:

```java
void AfterDestroyWindow(EventObject e)
```

which is your event handler.

After creating an instance of the VPE Control, you need to set the event listener (if an event listener is required):

```java
Doc = new VpeControl();
Doc.setVpeListener(new MyApplicationListener(this));
```

The VPE Control also implements an adapter class to test events:

```java
public class VpeEventTestAdapter implements VpeEventListener
```

This class dumps the names of occurring events to System.out. To use this class, implement your event listener as follows:

```java
class MyApplicationListener extends VpeEventTestAdapter
```

For practical examples, please see the Java demo source codes, which are installed together with VPE.
3.2 AfterDestroyWindow Event - Java

[GUI version only]
Is fired when the preview window was destroyed - for example closed by the user - and AutoDelete is True (the default). The document is also closed (removed from memory).

```java
public void AfterDestroyWindow(
    EventObject e
)

EventObject e
    the event object, see Java documentation

Remarks:
Do not call any VPE method or property when processing this event. The document is already closed and not accessible.

This event is fired, if AutoDelete is True and the user closes the preview. Otherwise the event AfterCloseWindow is fired.
3.3 Closing Event - Java

[GUI version only]

VPE requests confirmation from your application, if the preview can be closed.

```java
public void Closing(
    CancelEvent e
)
```

*CancelEvent e*

The cancel event is defined as:

```java
public class CancelEvent extends java.util.EventObject {
    private boolean cancel = false;

    public boolean getCancel() {
        return cancel;
    }

    public void setCancel(boolean cancel) {
        this.cancel = cancel;
    }

    public CancelEvent(VpeControl source) {
        super(source);
    }
}
```

You can call `e.setCancel(true)` if you want to cancel the event, i.e. deny that the preview is closed

**Example:**

```java
public void Closing(CancelEvent e) {
    if (!myDataIsSaved) {
        e.setCancel(true);
        JOptionPane.showMessageDialog(frame, "You must save first.");
    }
    else {
        e.setCancel(false);
        JOptionPane.showMessageDialog(frame, "Goodbye.");
    }
}
```
3.4 AfterCloseWindow Event - Java

[GUI version only]

Is fired, when the preview window was closed - for example by the user - and AutoDelete is False (which means, that the document is not destroyed, if the preview is closed).

```
public void AfterCloseWindow(
    EventObject e
)
```

*EventObject e*

the event object, see Java documentation

See also:

AfterDestroyWindow Event - Java
3.5 BeforeOpenFile Event - Java

[GUI version only]
Is fired when the user clicked the Open File button in the toolbar (or pushed the corresponding key).

```java
public void BeforeOpenFile(
    CancelEvent e
)
```

`CancelEvent e`

The cancel event is defined as:

```java
public class CancelEvent extends java.util.EventObject {
    private boolean cancel = false;

    public boolean getCancel() {
        return cancel;
    }

    public void setCancel(boolean cancel) {
        this.cancel = cancel;
    }

    public CancelEvent(VpeControl source) {
        super(source);
    }
}
```

You can call `e.setCancel(true)` if you want to cancel the event, i.e. deny that the file open dialog will be shown. Cancelling the operation allows you to display your own open dialog and to import or create your own document, for example from a database via a memory stream.

Example:

```java
public void BeforeOpenFile(CancelEvent e) {
    e.setCancel(true);
}
```

See also:

OpenFileName
3.6 AfterOpenFile Event - Java

[GUI version only]
Is fired after the file open operation has been completed.

```java
public void AfterOpenFile(
    EventObject e
)
```

*EventObject e*
the event object, see Java documentation

Remarks:
When handling this event, your application can acquire the error status of the file open operation by reading the property `LastError`. For example, if the user clicked onto the Cancel-Button in the open file dialog, `LastError` will be `ErrorCode.Cancelled`.

See also:
- `OpenFileName`
3.7 BeforeSaveFile Event - Java

[GUI version only]

Is fired when the user clicked the Save File button in the toolbar (or pushed the corresponding key).

```java
public void BeforeSaveFile(
    CancelEvent e
)
```

`CancelEvent e`

The cancel event is defined as:

```java
public class CancelEvent extends java.util.EventObject {
    private boolean cancel = false;

    public boolean getCancel() {
        return cancel;
    }

    public void setCancel(boolean cancel) {
        this.cancel = cancel;
    }

    public CancelEvent(VpeControl source) {
        super(source);
    }
}
```

You can call `e.setCancel(true)` if you want to cancel the event, i.e. deny that the file save dialog will be shown. Cancelling the operation allows you to display your own save dialog and to export your own document, for example to a memory stream and from there to a database.

**Example:**

```java
public void BeforeSaveFile(CancelEvent e) {
    e.setCancel(true);
}
```

**See also:**

- `SaveFileName`
3.8 AfterSaveFile Event - NET

[GUI version only]

Is fired after the file save operation has been completed.

public void AfterSaveFile(
    EventObject e
)

Remarks:

When handling this event, your application can acquire the error status of the file open operation by reading the property LastError. For example, if the user clicked onto the Cancel-Button in the save file dialog, LastError will be ErrorCode.Cancelled.

See also:

   SaveFileName
3.9 HelpRequested Event - Java

[GUI version only]

Is sent if the property RouteHelp is True and the user clicked the Help-Button in the toolbar or pushed the corresponding key (F1 by default).

This is one of the few events where CloseDoc() may be called while processing it.

```java
public void HelpRequested(
    EventObject e
)
```

`EventObject e`
the event object, see Java documentation
3.10 AfterAutoPageBreak Event - Java

Is sent, when a Auto Page Break occurred (see "Automatic Text Break" in the Programmer's Manual). The engine is on the new page. This is either a newly generated page, or an already existing page (if the text was inserted on a page, that has already pages following).

This gives you additional control over the layout. When the event is fired, VPE already moved to the next page (or generated one) and you can modify the Output Rectangle (NOT the Default Output Rectangle) to control the layout, or insert manually Headers and Footers, etc. (see "Headers and Footers" in the Programmer's Manual).

```java
public void AfterAutoPageBreak(
    EventObject e
)

EventObject e
the event object, see Java documentation

Remarks:
If your application is not able to receive events, or if you don’t want to process this event for some reason, you can test for AutoBreak (this means, check if an AutoBreak occurred after inserting a text object) with the following technique:

Store the current page-number in a variable. After the output, compare this variable to the current page-number. If it is different, an AutoBreak had occurred.

Code example:
```
n = Doc.CurrentPage  // remember the current page number
Doc.Print(1, 1, "... very long text ...")
if n <> Doc.CurrentPage
    caught auto break!
    The formula "Doc.CurrentPage - n" returns the number of automatically generated pages.
```

Notes:
If you are inserting text objects into the document while processing this event, make sure they will not cause again an Auto Page Break event - otherwise this will cause an endless recursion, ending up in a stack overflow with a GPF.

To avoid Auto Page Breaks, set AutoBreakMode = AUTO_BREAK_NO_LIMITS.

If you are modifying any properties, as for example the AutoBreakMode or FontSize etc., it will not be reset when you code exits the event handler. If you are changing properties, you need to save and restore them by code yourself. Do this by calling StoreSet() in the beginning of you event handler and by calling UseSet() followed by RemoveSet() at the end.

Example: if a Print() statement in your main code causes an AutoBreak and if your AutoBreak Handler sets the PenSize to something different than zero, then the rest of the text which is output by the previous Print() command of your main code will have a box around it.
Solution: at the very first beginning of your AutoBreak Handler call StoreSet() and on the last line call UseSet() and then RemoveSet(). Do this only if required, you can also check which single properties have been changed and change them back, because StoreSet() and UseSet() eat performance!

YOU MAY NOT USE NoHold of the embedded flags in the AutoBreak-Event Handler. This will cause an internal recursion and overwrite settings done in the main code. Explanation: When using the flag NoHold in your main code in a Print(Box)() or Write(Box)() statement which
causes an AutoBreak, the current settings are stored in a temporary memory block (to be restored after the command has been executed). Afterwards your AutoBreak Event-Handler is executed and if you use the NoHold Flag again, the temporary memory block is overwritten, so that its initial settings are lost.
### 3.11 RequestPrint Event - Java

[GUI version only]

Is fired to inform the application about the several stages during the printing process.

```java
public void RequestPrint(
    RequestPrintEvent e
)
```

**RequestPrintEvent e**

The request print event is defined as:

```java
public class RequestPrintEvent extends java.util.EventObject {
    private int printAction;
    private int printResultingAction = PrintResultingAction.Ok;

    public void setPrintResultingAction(int printResultingAction) {
        this.printResultingAction = printResultingAction;
    }

    public int getPrintResultingAction() {
        return printResultingAction;
    }

    public int getPrintAction() {
        return printAction;
    }

    public RequestPrintEvent(VpeControl source, int printAction) {
        super(source);
        this.printAction = printAction;
    }
}
```

*printAction* has one of the following values:

```java
public class PrintAction {
    public static final Abort = 0; // User aborted while printing
    public static final Start = 1; // Print started
    public static final End = 2; // Print ended
    public static final SetupAbort = 3; // User aborted Setup-Dialog
    public static final SetupStart = 4; // Setup-Dialog started
    public static final SetupEnd = 5; // Setup-Dialog ended
}
```

*printResultingAction* is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

```java
public class PrintResultingAction {
    public static final Ok = 0;
    public static final Abort = 1;
}
```

**Remarks:**

Do not call `CloseDoc()` while processing this event. You would terminate a module that is working.
When your application handles this event, the printAction member contains the current status of the printing progress. Your application’s event handler can return a value in the printResultingAction member.

Your application should return **Ok** if it processes this message, except for PrintAction = SetupStart, where your application may return in addition **Abort** to abort the print job. SetupStart is sent, when the user clicked the print button in the preview (or pushed the corresponding key). You have the option to abort the job, for you can then create and print internally a new document which is completely different to the preview.

Another use for the SetupStart message is, to pre-initialize the **Device Control Properties** at this stage, before the printer setup dialog will be shown to the user.
3.12 BeforePrintNewPage Event - Java

[GUI version only]

Is fired only while printing exactly before printing a new page. The event is useful to change the Device Control Properties on-the-fly during printing. You may change all properties, except the device itself.

```java
public void BeforePrintNewPage(
    PrintEvent e
)
```

*PrintEvent e*

The print event is defined as:

```java
class PrintEvent extends java.util.EventObject {
    private int page;
    private int printResultingAction=PrintResultingAction.Ok;

    public void setPrintResultingAction(int printResultingAction) {
        this.printResultingAction = printResultingAction;
    }

    public int getPrintResultingAction() {
        return printResultingAction;
    }

    public int getPage() {
        return page;
    }

    PrintEvent(VpeControl source, int page) {
        super(source);
        this.page = page;
    }
}
```

*int page*

current page number that will be printed

*printResultingAction* is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

```java
class PrintResultingAction {
    public static final Ok = 0;
    public static final Change = 1;
}
```

**Remarks:**

Your application’s event handler can return a value in the printResultingAction member of the PrintEvent parameter.

Your application should return **Ok**, if it processes this message without changing a printing device’s properties.

If a Device Control Property was changed, your application must return **Change**.
If you change the *Device Control Properties* during the print job, you must reset them to the original values, when the print job has finished (see RequestPrint Event – Java: with printAction = Abort or printAction = End).

Changing the properties (like DevPaperBin, DevOrientation, DevPaperFormat, etc.) during the print job doesn’t work with some (buggy) printer drivers, for example some fax drivers and the HP4 M Plus driver (the HP4 PS driver should work with the HP4 M Plus printer!).

Some properties and methods don’t work with some printer drivers. For example "DevTTOption" doesn’t work with our HP4 and HP5 printer drivers on WfW 3.11 and NT 3.51, but it works with both drivers on Win95. This is a driver problem.

*Printer drivers are manufactured by vendors independent of IDEAL Software; we make no warranty, implied or otherwise, regarding these product's performance or reliability.*

**Known Problems:**

**Changing any Device Control Properties during the print job disables Duplex Printing on PCL Printers**

**SYMPTOMS**

When you change a Device Control Property during the print job, it appears to disable Duplex (double-sided) printing when the target printer is a (Hewlett Packard) PCL printer.

**CAUSE**

PCL printers treat a change in paper size as a new print job that requires the printer to be initialized. This causes the printer to eject any page that is currently in the printer. The PCL printer drivers for Windows assume that the page size has been changed when a Device Control Property is changed during the print job, unless the orientation of the page has changed.

**RESOLUTION**

To prevent having a page ejected when changing a Device Control Property during the print job, make sure that the function is called only between individual sheets of paper. Changing a Device Control Property before printing odd-numbered pages is sufficient for most applications that use duplex printing. However, some applications require that you change the page orientation on a page-by-page basis. In this case, you can change Device Control Properties between individual sheets of paper if the orientation has changed.

**STATUS**

This behavior is by design.

**MORE INFORMATION**

Note that when this problem occurs the print job continues and the sheets of paper are passed through the printer's duplexer, but the sheets are only printed on one side.

Because of the page size initialization requirement for PCL printers, Windows PCL drivers treat the change of Device Control Properties differently. These drivers allow only the orientation to change between the front and back pages of a sheet of paper. This means that the change of a Device Control Property will eject the page unless the orientation (and only the orientation) has changed from the previous page. Returning `PrintResultingAction.Change` although no Device Control Property has been changed causes the printer to eject the page.
Returning `PrintResultingAction.Change` although no Device Control Property has been changed is unnecessary. By doing so, you risk having a page ejected from the printer, which has a high probability of occurring.

**See also:**

- `PrintDevData Event - Java`
3.13 PrintDevData Event - Java

[GUI version only]

Is sent only while printing, exactly before printing a new page and immediately after
BeforePrintNewPage() has been sent. The only use for this event is to call DevSendData() in
response.

DevSendData() enables your application to send escape sequences to the printing device. So it is
possible to select for example an output paper bin by code (or whatever other functionality is
provided by the connected printer).

public void PrintDevData(
    PrintEvent e
)

PrintEvent e

The print event is defined as:

```java
public class PrintEvent extends java.util.EventObject {
    private int page;
    private int printResultingAction=PrintResultingAction.Ok;

    public void setPrintResultingAction(int printResultingAction) {
        this.printResultingAction = printResultingAction;
    }

    public int getPrintResultingAction() {
        return printResultingAction;
    }

    public int getPage() {
        return page;
    }

    public PrintEvent(VpeControl source, int page) {
        super(source);
        this.page = page;
    }
}
```

- **int page**
  current page number that will be printed

- **printResultingAction** is a return-parameter, i.e. you can assign it one of the following values to
  control the resulting action of VPE:

```java
public class PrintResultingAction {
    public static final Ok = 0;
    public static final Change = 1;
}
```

**Remarks:**

When your application handles this event, the PrintAction member contains the current status
of the printing progress. Your application’s event handler can return a value in the
PrintResultingAction member.

Your application should return Ok if it processes this message without calling DevSendData().
If it called DevSendData(), your application must return **Change**.

**See also:**

- BeforePrintNewPage Event
3.14 BeforeMail Event - Java

[GUI version only]
Is fired, when the user clicked the eMail-button in the preview (or pushed the corresponding key). The e-mail was not sent yet, therefore your application has now the option to set receivers, attachments, etc. by code.

```java
public void BeforeMail(
    CancelEvent e
)
```

The cancel event is defined as:

```java
public class CancelEvent extends java.util.EventObject {
    private boolean cancel = false;

    public boolean getCancel() {
        return cancel;
    }

    public void setCancel(boolean cancel) {
        this.cancel = cancel;
    }

    public CancelEvent(VpeControl source) {
        super(source);
    }
}
```

You can call `e.setCancel(true)` if you want to cancel the event, i.e. deny that the document will be mailed. Cancelling the operation allows you to display your own mail dialog and/or to execute your own mailing code.

Example:

```java
public void BeforeMail(CancelEvent e) {
    e.setCancel(true);
    // Here comes your own code to mail a document
}
```

See also:

E-Mail Functions
3.15 AfterMail Event - Java

[GUI version only]

Is sent, after the user clicked the eMail-button in the preview (or pushed the corresponding key) and the e-mail has already been sent.

```java
public void AfterMail(
    EventObject e
)
```

*EventObject e*
the event object, see Java documentation

**Remarks:**
When handling this event, your application can acquire the error status of the mail by reading the property **LastError**.

**See also:**
- E-Mail Functions
3.16 ObjectClicked Event - Java

[GUI version only, Professional Edition and above]

A clickable object with an assigned ObjectID has been clicked with the mouse. This is one of the few events where CloseDoc() may be called while processing the event.

```java
public void ObjectClicked(
    ObjectClickedEvent e
)
```

*ObjectClickedEvent e*

The object clicked event is defined as:

```java
public class ObjectClickedEvent extends java.util.EventObject {
    private int objectID;

    public int getObjectID() {
        return objectID;
    }

    public ObjectClickedEvent(VpeControl source, int ObjectID) {
        super(source);
        this.objectID = ObjectID;
    }
}
```

objectId is the Object ID of the object that was clicked by the user

*See also:*

Clickable objects
3.17 AfterControlEnter - Java
[GUI version only, Interactive Edition and above]
A Control received the focus.
Normally, this event is sent if the user presses the Tab or Backtab key, or clicks with the mouse onto a control. This event can be used to re-format the content of a control.

```java
public void AfterControlEnter(
    VPEObjectEvent e
)
```

_VPEObjectEvent e_

The VPE object event is defined as:

```java
public class VPEObjectEvent extends java.util.EventObject{
  private TVPEObject VpeObject;

  public TVPEObject getVpeObject() {
    return VpeObject;
  }

  public VPEObjectEvent(VpeControl source, TVPEObject VpeObject) {
    super(source);
    this.VpeObject = VpeObject;
  }
}
```

VpeObject is the VPE object that fired the event

See also:
Interactive Documents
3.18 RequestControlExit - Java

[GUI version only, Interactive Edition and above]

The user wishes to remove the focus from the currently focused Control. In response to this event your application can evaluate the value of the Control and decide, whether the current value is valid and the Control may lose the focus or not.

```java
public void RequestControlExit(
    ControlExitEvent e
)
```

*ControlExitEvent e*

The control exit event is defined as:

```java
public class ControlExitEvent extends VPEObjectEvent {
    private boolean canExit = true;

    public boolean getCanExit() {
        return canExit;
    }

    public void setCanExit(boolean canExit) {
        this.canExit = canExit;
    }

    public ControlExitEvent(VpeControl source, TVPEObject VpeObject) {
        super(source, VpeObject);
    }
}
```

VpeObject is the VPE object that fired the event

*boolean CanExit*

is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>yes, the control may lose the focus</td>
</tr>
<tr>
<td>false</td>
<td>no, the control may not lose the focus</td>
</tr>
</tbody>
</table>

See also:

Interactive Documents
3.19 AfterControlExit - Java

[GUI version only, Interactive Edition and above]

The currently focused Control lost the focus.
Normally, this event is sent if the user presses the Tab or Backtab key, or clicks with the mouse onto another control. When receiving this message, it is possible for you to force the focus to be set explicitly to a specific control by calling SetFocusControlByName() or SetFocus(). It is also possible to enable and disable other controls of the current form while processing this event. In addition this event can be used to re-format the content of a control.

```java
public void AfterControlExit(
    VPEObjectEvent e
)
```

VPEObjectEvent e
The VPE object event is defined as:

```java
public class VPEObjectEvent extends java.util.EventObject{
    private TVPEObject VpeObject;

    public TVPEObject getVpeObject() {
        return VpeObject;
    }

    public VPEObjectEvent(VpeControl source, TVPEObject VpeObject) {
        super(source);
        this.VpeObject = VpeObject;
    }
}
```

VpeObject is the VPE object that fired the event

See also:
Interactive Documents
3.20 AfterControlChange - Java

[GUI version only, Interactive Edition and above]

A Control changed its value, i.e. the content of a Control was edited by the user or the value of an associated Field was changed by code.
Evaluating this event means, that your application is informed about every single keystroke or mouse-click, which modifies a Control's content.
If you are working with Fields that are associated with controls - as recommended - your application should not take care of this event.
This is one of the few events where CloseDoc() may be called while processing the event.
The event is not fired, if you set the value of a Control by code.

```java
public void AfterControlChange(VPEObjectEvent e)
```

VPEObjectEvent e
The VPE object event is defined as:
```java
public class VPEObjectEvent extends java.util.EventObject{
    private TVPEObject VpeObject;

    public TVPEObject getVpeObject() {
        return VpeObject;
    }

    public VPEObjectEvent(VpeControl source, TVPEObject VpeObject) {
        super(source);
        this.VpeObject = VpeObject;
    }
}
```
VpeObject is the VPE object that fired the event

Remarks:
If you change the value of a Control by code, the AfterControlChange event is not fired. But if the Control is associated with a Field, the event AfterFieldChange is fired.
Vice versa, if you change the value of a Field by code, the AfterFieldChange event is not fired, but any Controls associated with the Field will fire the event AfterControlChange.

See also:
Interactive Documents
3.21 AfterFieldChange - Java

[GUI version only, Interactive Edition and above]

A Field (not a control!) changed its value, because the associated Control was edited by the user or the content of any associated Control was changed by code. This event is very interesting, because a Field will change its value each time the user makes a change. Evaluating this event means, that your application is informed about every single keystroke or mouse-click, which modifies a Control's content. This is one of the few events where CloseDoc() may be called while processing the event. The event is not fired, if you set the value of a Field by code.

```java
public void AfterFieldChange(
    VPEFieldEvent e
)
```

VPEFieldEvent e

The VPE field event is defined as:

```java
public class VPEFieldEvent extends java.util.EventObject {
    private TVPEField VpeField;

    public TVPEField getVpeField() {
        return VpeField;
    }

    public VPEFieldEvent(VpeControl source, TVPEField VpeField) {
        super(source);
        this.VpeField = VpeField;
    }
}
```

VpeField is the Field object that fired the event

Remarks:

If you change the value of a Control by code, the AfterControlChange event is not fired. But if the Control is associated with a Field, the event AfterFieldChange is fired. Vice versa, if you change the value of a Field by code, the AfterFieldChange event is not fired, but any Controls associated with the Field will fire the event AfterControlChange.

See also:

Interactive Documents
4 Events Generated by VPE - ActiveX

4.1 Events Generated by VPE - ActiveX

The VPE-ActiveX fires several events to your application, so you have always total control about what's happening.

NOTE: You may not call CloseDoc() while processing any event fired by VPE - except it is explicitly said in the description of a particular event that it is allowed.
4.2 AfterDestroyWindow Event

Is fired when the preview window was destroyed - for example closed by the user - and AutoDelete is True (the default). The document is also closed (removed from memory).

Remarks:

Do not call any VPE method or property when processing this event. The document is already closed and not accessible.

This event is fired, if AutoDelete is True and the user closes the preview. Otherwise the event AfterCloseWindow is fired.
4.3 RequestClose Event

VPE requests confirmation from your application, if the preview can be closed.

OnRequestClose(
   CanClose as Boolean
)

CanClose
   is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>no, the preview may not be closed</td>
</tr>
<tr>
<td>True</td>
<td>Otherwise</td>
</tr>
</tbody>
</table>
4.4 AfterCloseWindow Event

Is fired, when the preview window was closed - for example by the user - and AutoDelete is False (which means, that the document is not destroyed if the preview is closed).

See also:
   AfterDestroyWindow Event
4.5 BeforeOpenFile Event

Is fired when the user clicked the Open File button in the toolbar (or pushed the corresponding key).

BeforeOpenFile(
       Cancel as Boolean
   )

Cancel
is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>continue operation</td>
</tr>
<tr>
<td>True</td>
<td>cancel the event, i.e. deny that the file open dialog will be shown. Cancelling the operation allows you to display your own open dialog and to import or create your own document, for example from a database via a memory stream.</td>
</tr>
</tbody>
</table>

Example:

```vba
Protected Sub vpe_BeforeOpenFile (Cancel as Boolean)
    Cancel = True
End Sub
```

See also:

OpenFileName
4.6 AfterOpenFile Event

Is fired after the file open operation has been completed.

```vbnet
AfterOpenFile(
    ByVal Result As Long
)
```

*Result*

the status of the operation, one of the VERR_xyz error codes. For example, if the user clicked onto the Cancel-Button in the open file dialog, *Result* will be VERR CANCELLED.

**See also:**

- `OpenFileName`
4.7 BeforeSaveFile Event

Is fired when the user clicked the Save File button in the toolbar (or pushed the corresponding key).

BeforeSaveFile(
    Cancel as Boolean
)

Cancel
is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>continue operation</td>
</tr>
<tr>
<td>True</td>
<td>cancel the event, i.e. deny that the file save dialog will be shown. Cancelling the operation allows you to display your own save dialog and to export your own document, for example to a memory stream and from there to a database.</td>
</tr>
</tbody>
</table>

Example:

```vpe
Protected Sub vpe_BeforeSaveFile (Cancel as Boolean)
    Cancel = True
End Sub
```

See also:

SaveFileName
4.8 AfterSaveFile Event
Is fired after the file save operation has been completed.

AfterSaveFile(
    ByVal Result As Long
)

Result
the status of the operation, one of the VERR_xyz error codes. For example, if the user clicked onto the Cancel-Button in the save file dialog, Result will be VERR_CANCELED.

See also:
SaveFileName
4.9 DoHelp Event

Is sent if the property RouteHelp is True and the user clicked the Help-Button in the toolbar or pushed the corresponding key (F1 by default).

This is one of the few events where CloseDoc() may be called while processing it.
4.10 AfterAutoPageBreak Event

Is sent, when a Auto Page Break occurred (see "Automatic Text Break" in the Programmer's Manual). The engine is on the new page. This is either a newly generated page, or an already existing page (if the text was inserted on a page, that has already pages following).

This gives you additional control over the layout. When the event is fired, VPE already moved to the next page (or generated one) and you can modify the Output Rectangle (NOT the Default Output Rectangle) to control the layout, or insert manually Headers and Footers, etc. (see "Headers and Footers" in the Programmer's Manual).

Remarks:
If your application is not able to receive events, or if you don't want to process this event for some reason, you can test for AutoBreak (this means, check if an AutoBreak occurred after inserting a text object) with the following technique:

Store the current page-number in a variable. After the output, compare this variable to the current page-number. If it is different, an AutoBreak had occurred.

Code example:

```vbnet
n = Doc.CurrentPage // remember the current page number
Doc.Print(1, 1, "... very long text ...")
if n <> Doc.CurrentPage
    ' caught auto break!
    The formula "Doc.CurrentPage - n" returns the number of automatically generated pages.
```

Notes:
If you are inserting text objects into the document while processing this event, make sure they will not cause again an Auto Page Break event - otherwise this will cause an endless recursion, ending up in a stack overflow with a GPF.

To avoid Auto Page Breaks, set AutoBreakMode = AUTO_BREAK_NO_LIMTS.

If you are modifying any properties, as for example the AutoBreakMode or FontSize etc., it will not be reset when you code exits the event handler. If you are changing properties, you need to save and restore them by code yourself. Do this by calling StoreSet() in the beginning of you event handler and by calling UseSet() followed by RemoveSet() at the end.

Example: if a Print() statement in your main code causes an AutoBreak and if your AutoBreak Handler sets the PenSize to something different than zero, then the rest of the text which is output by the previous Print() command of your main code will have a box around it.

Solution: at the very first beginning of your AutoBreak Handler call StoreSet() and on the last line call UseSet() and then RemoveSet(). Do this only if required, you can also check which single properties have been changed and change them back, because StoreSet() and UseSet() eat performance!

YOU MAY NOT USE NoHold of the embedded flags in the AutoBreak-Event Handler. This will cause an internal recursion and overwrite settings done in the main code. Explanation: When using the flag NoHold in your main code in a Print(Box)() or Write(Box)() statement which causes an AutoBreak, the current settings are stored in a temporary memory block (to be restored after the command has been executed). Afterwards your AutoBreak Event-Handler is executed and if you use the NoHold Flag again, the temporary memory block is overwritten, so that its initial settings are lost.
4.11 RequestPrint Event - ActiveX

Is fired to inform the application about the several stages during the printing process.

RequestPrint(
   ByVal Action As Long,
   ResultingAction As Long
)

<table>
<thead>
<tr>
<th>Action</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT_MSG_ABORT</td>
<td>0</td>
<td>User aborted while printing</td>
</tr>
<tr>
<td>PRINT_MSG_START</td>
<td>1</td>
<td>Print started</td>
</tr>
<tr>
<td>PRINT_MSG_END</td>
<td>2</td>
<td>Print ended</td>
</tr>
<tr>
<td>PRINT_MSG_SETUPABORT</td>
<td>3</td>
<td>User aborted Setup-Dialog</td>
</tr>
<tr>
<td>PRINT_MSG_SETUPSTART</td>
<td>4</td>
<td>Setup-Dialog started</td>
</tr>
<tr>
<td>PRINT_MSG_SETUPEnd</td>
<td>5</td>
<td>Setup-Dialog ended</td>
</tr>
</tbody>
</table>

ResultingAction

is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Resulting Action</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT_ACTION_OK</td>
<td>0</td>
</tr>
<tr>
<td>PRINT_ACTION_ABORT</td>
<td>1</td>
</tr>
</tbody>
</table>

Remarks:

Do not call CloseDoc() while processing this event. You would terminate a module that is working.

Your application should return PRINT_ACTION_OK (zero) if it processes this message, except for Action = PRINT_MSG_SETUPSTART, where your application may return in addition PRINT_ACTION_ABORT (= 1) to abort the print job. PRINT_MSG_SETUPSTART is sent, when the user clicked the print button in the preview (or pushed the corresponding key). You have the option to abort the job, for you can then create and print internally a new document which is completely different to the preview.

Another use for the PRINT_MSG_SETUPSTART message is, to pre-initialize the Device Control Properties at this stage, before the printer setup dialog will be shown to the user.
4.12 BeforePrintNewPage Event

Is fired only while printing exactly before printing a new page. The event is useful to change the Device Control Properties on-the-fly during printing. You may change all properties, except the device itself.

BeforePrintNewPage(  
    ByVal Page As Long,  
    long ResultingAction  
)

Page  
current page number that will be printed

ResultingAction  
is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Resulting Action</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT_ACTION_OK</td>
<td>0</td>
</tr>
<tr>
<td>PRINT_ACTION_ABORT</td>
<td>1</td>
</tr>
</tbody>
</table>

Remarks:
Your application should return PRINT_ACTION_OK (zero) if it processes this message without changing a printing device’s properties.

If a Device Control Property was changed, your application must return PRINT_ACTION_CHANGE (= 1).

- If you change the Device Control Properties during the print job, you must reset them to the original values, when the print job has finished (see RequestPrint Event – ActiveX: with Action = PRINT_MSG_ABORT or Action = PRINT_MSG_END).
- Changing the properties (like DevPaperBin, DevOrientation, DevPaperFormat, etc.) during the print job doesn’t work with some (buggy) printer drivers, for example some fax drivers and the HP4 M Plus driver (the HP4 PS driver should work with the HP4 M Plus printer!).
- Some properties and methods don’t work with some printer drivers. For example "DevTTOption" doesn’t work with our HP4 and HP5 printer drivers on WfW 3.11 and NT 3.51, but it works with both drivers on Win95. This is a driver problem.
- Printer drivers are manufactured by vendors independent of IDEAL Software; we make no warranty, implied or otherwise, regarding these product’s performance or reliability.
- Win32s is not officially supported by VPE. The Device Control Properties do not work under Win32s.

Known Problems:
Changing any Device Control Properties during the print job disables Duplex Printing on PCL Printers

SYMPTOMS
When you change a Device Control Property during the print job, it appears to disable Duplex (double-sided) printing when the target printer is a (Hewlett Packard) PCL printer.

**CAUSE**

PCL printers treat a change in paper size as a new print job that requires the printer to be initialized. This causes the printer to eject any page that is currently in the printer. The PCL printer drivers for Windows assume that the page size has been changed when a Device Control Property is changed during the print job, unless the orientation of the page has changed.

**RESOLUTION**

To prevent having a page ejected when changing a Device Control Property during the print job, make sure that the function is called only between individual sheets of paper. Changing a Device Control Property before printing odd-numbered pages is sufficient for most applications that use duplex printing. However, some applications require that you change the page orientation on a page-by-page basis. In this case, you can change Device Control Properties between individual sheets of paper if the orientation has changed.

**STATUS**

This behavior is by design.

**MORE INFORMATION**

Note that when this problem occurs the print job continues and the sheets of paper are passed through the printer's duplexer, but the sheets are only printed on one side.

Because of the page size initialization requirement for PCL printers, Windows PCL drivers treat the change of Device Control Properties differently. These drivers allow only the orientation to change between the front and back pages of a sheet of paper. This means that the change of a Device Control Property will eject the page unless the orientation (and only the orientation) has changed from the previous page. Returning PRINT_ACTION_CHANGE although no Device Control Property has been changed causes the printer to eject the page.

Returning PRINT_ACTION_CHANGE although no Device Control Property has been changed is unnecessary. By doing so, you risk having a page ejected from the printer, which has a high probability of occurring.

**See also:**

DoPrintDevData Event
4.13 DoPrintDevData Event

Is sent only while printing, exactly before printing a new page and immediately after BeforePrintNewPage() has been sent. The only use for this event is to call DevSendData() in response.

DevSendData enables your application to send escape sequences to the printing device. So it is possible to select for example an output paper bin by code (or whatever other functionality is provided by the connected printer).

**DoPrintDevData**(
  ByVal Page As Long,
  long ResultingAction
)

*Page*
  current page number that will be printed

*ResultingAction*
  is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Resulting Action</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT_ACTION_OK</td>
<td>0</td>
</tr>
<tr>
<td>PRINT_ACTION_ABORT</td>
<td>1</td>
</tr>
</tbody>
</table>

**Remarks:**
Your application should return PRINT_ACTION_OK (zero) if it processes this message without calling DevSendData().

If it called DevSendData(), your application must return PRINT_ACTION_CHANGE (= 1).

**See also:**
BeforePrintNewPage Event
4.14 BeforeMail Event

Is fired, when the user clicked the eMail-button in the preview (or pushed the corresponding key). The e-mail was not sent yet, therefore your application has now the option to set receivers, attachments, etc. by code.

BeforeMail(
    Cancel As Boolean
)

Cancel
is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>cancel operation i.e. do not mail the document</td>
</tr>
<tr>
<td>False</td>
<td>continue operation</td>
</tr>
</tbody>
</table>

If you set this parameter to True, the operation is cancelled, i.e. VPE will not mail the document. Cancelling the operation allows you to display your own mail dialog and/or to execute your own mailing code.

See also:
E-Mail Functions
4.15 AfterMail Event

Is sent, after the user clicked the eMail-button in the preview (or pushed the corresponding key) and the e-mail has already been sent.

```vba
AfterMail(
    ByVal Result As Long
)
```

*Result*

the status of the e-mail action, one of the VERR_xyz error codes

See also:

- E-Mail Functions
4.16 DoObjectClicked Event

[Professional Edition and above]

A clickable object with an assigned ObjectId has been clicked with the mouse.
This is one of the few events where CloseDoc() may be called while processing the event.

DoObjectClicked(
   ByVal ObjectId As Long
)

ObjectId
   the Object ID of the object that was clicked by the user

See also:
   Clickable Objects
4.17 DoUDOPaint Event

[Professional Edition and above]

Is fired as a notification from a User Defined Object (UDO). The object needs to be painted to the output device.

See the description of the User Defined Objects for information on how to process this event.

DoUDOPaint(
)

See also:

User Defined Objects
4.18 AfterControlEnter

[Interactive Edition and above]

A Control received the focus.
Normally, this event is sent if the user presses the Tab or Backtab key, or clicks with the mouse onto a control. This event can be used to re-format the content of a control.

AfterControlEnter(
    ByVal Obj As TVPEObject
)

Obj
    the object, which fired the event

See also:
    Interactive Documents
4.19 RequestControlExit

[Interactive Edition and above]

The user wishes to remove the focus from the currently focused Control. In response to this event your application can evaluate the value of the Control and decide, whether the current value is valid and the Control may lose the focus or not.

RequestControlExit(
   ByVal Obj As TVPEObject,
   CanExit As Boolean
)

Obj
   the object, which fired the event

CanExit
   is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, the control may loose the focus</td>
</tr>
<tr>
<td>False</td>
<td>no, the control may not loose the focus</td>
</tr>
</tbody>
</table>

See also:
Interactive Documents
4.20 AfterControlExit

[Interactive Edition and above]

The currently focused Control lost the focus. Normally, this event is sent if the user presses the Tab or Backtab key, or clicks with the mouse onto another control. When receiving this message, it is possible for you to force the focus to be set explicitly to a specific control by calling `SetFocusControlByName()` or `SetFocus()`. It is also possible to enable and disable other controls of the current form while processing this event. In addition this event can be used to re-format the content of a control.

```vpe
AfterControlExit(
    ByVal Obj As TVPEObject
)
```

`Obj` the object, which fired the event

See also:
- Interactive Documents
4.21 AfterControlChange

[Interactive Edition and above]

A Control changed its value, i.e. the content of a Control was edited by the user or the value of an associated Field was changed by code.
Evaluating this event means, that your application is informed about every single keystroke or mouse-click, which modifies a Control's content.
If you are working with Fields that are associated with controls - as recommended - your application should not take care of this event.
This is one of the few events where CloseDoc() may be called while processing the event.
The event is not fired, if you set the value of a Control by code.

AfterControlChange(
  ByVal Obj As TVPEObject
)

Obj
  the object, which fired the event

Remarks:
  If you change the value of a Control by code, the AfterControlChange event is not fired. But if the Control is associated with a Field, the event AfterFieldChange is fired.
  Vice versa, if you change the value of a Field by code, the AfterFieldChange event is not fired, but any Controls associated with the Field will fire the event AfterControlChange.

See also:
  Interactive Documents
4.22 AfterFieldChange

[Interactive Edition and above]

A Field (not a control!) changed its value, because the associated Control was edited by the user or the content of any associated Control was changed by code. This event is very interesting, because a Field will change its value each time the user makes a change. Evaluating this event means, that your application is informed about every single keystroke or mouse-click, which modifies a Control's content. This is one of the few events where CloseDoc() may be called while processing the event. The event is not fired, if you set the value of a Field by code.

AfterFieldChange(
    ByVal Obj As TVPEField
)

Obj the Field object, which fired the event

Remarks:
If you change the value of a Control by code, the AfterControlChange event is not fired. But if the Control is associated with a Field, the event AfterFieldChange is fired. Vice versa, if you change the value of a Field by code, the AfterFieldChange event is not fired, but any Controls associated with the Field will fire the event AfterControlChange.

See also:
Interactive Documents
5 Events Generated by VPE - VCL

5.1 Events Generated by VPE - VCL

The VPE-VCL (TVPEngine) fires several events to your application, so you have always total control about what's happening.

NOTE: You may not call `CloseDoc()` while processing any event fired by VPE - except it is explicitly said in the description of a particular event that it is allowed.
5.2 OnDestroyWindow Event

Is fired when the preview window was destroyed - for example closed by the user - and AutoDelete is True (the default). The document is also closed (removed from memory).

OnDestroyWindow(
    Sender: TVP Engine)

Sender
    the VPE object that fired the event

Remarks:
    Do not call any VPE method or property when processing this event. The document is already closed and not accessible.

This event is fired, if AutoDelete is True and the user closes the preview. Otherwise the event OnCloseWindow is fired.
5.3 OnRequestClose Event

VPE requests confirmation from your application, if the preview can be closed.

OnRequestClose(
    Sender: TVPEngine;
    var CanClose: Boolean
)

Sender
the VPE object that fired the event

CanClose
is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>no, the preview may not be closed</td>
</tr>
<tr>
<td>True</td>
<td>Otherwise</td>
</tr>
</tbody>
</table>
5.4 OnCloseWindow Event

Is fired, when the preview window was closed - for example by the user - and AutoDelete is False (which means, that the document is not destroyed if the preview is closed).

OnCloseWindow(
    Sender: TVPEngine
)

Sender
the VPE object that fired the event

See also:
 OnDestroyWindow Event
5.5 OnBeforeOpenFile Event

Is fired when the user clicked the Open File button in the toolbar (or pushed the corresponding key).

OnBeforeOpenFile(
    Sender: TVPEngine;
    var Cancel: Boolean
)

Sender
the VPE object that fired the event

Cancel
is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>continue operation</td>
</tr>
<tr>
<td>True</td>
<td>cancel the event, i.e. deny that the file open dialog will be shown. Cancelling the operation allows you to display your own open dialog and to import or create your own document, for example from a database via a memory stream.</td>
</tr>
</tbody>
</table>

See also:
OpenFileName
5.6 OnAfterOpenFile Event

Is fired after the file open operation has been completed.

OnAfterOpenFile(
    Sender: TVPEngine;
    Result: Integer
)

Sender
    the VPE object that fired the event

Result
    the status of the operation, one of the VERR_xyz error codes. For example, if the user clicked onto the Cancel-Button in the open file dialog, Result will be VERR_CANCELED.

See also:
    OpenFileName
5.7 OnBeforeSaveFile Event

Is fired when the user clicked the Save File button in the toolbar (or pushed the corresponding key).

```pascal
OnBeforeSaveFile(
    Sender: TVPEngine;
    var Cancel: Boolean
)
```

*Sender*
the VPE object that fired the event

*Cancel*
is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>continue operation</td>
</tr>
<tr>
<td>True</td>
<td>cancel the event, i.e. deny that the file save dialog will be shown. Cancelling the operation allows you to display your own save dialog and to export your own document, for example to a memory stream and from there to a database.</td>
</tr>
</tbody>
</table>

See also:

*SaveFileName*
5.8 OnAfterSaveFile Event

Is fired after the file save operation has been completed.

OnAfterSaveFile(  
    Sender: TVPEngine;  
    Result: Integer  
)

Sender  
the VPE object that fired the event

Result  
the status of the operation, one of the VERR_xyz error codes. For example, if the user clicked onto the Cancel-Button in the save file dialog, Result will be VERR_CANCELLLED.

See also:  
SaveFileName
5.9 OnHelp Event

Is sent if the property RouteHelp is True and the user clicked the Help-Button in the toolbar or pushed the corresponding key (F1 by default).

This is one of the few events where CloseDoc() may be called while processing it.

OnHelp(
   Sender: TVPEngine
)

Sender
   the VPE object that fired the event
5.10 OnAutoPageBreak Event

Is sent, when a Auto Page Break occurred (see "Automatic Text Break" in the Programmer's Manual). The engine is on the new page. This is either a newly generated page, or an already existing page (if the text was inserted on a page, that has already pages following).

This gives you additional control over the layout. When the event is fired, VPE already moved to the next page (or generated one) and you can modify the Output Rectangle (NOT the Default Output Rectangle) to control the layout, or insert manually Headers and Footers, etc. (see "Headers and Footers" in the Programmer's Manual).

OnAutoPageBreak(
  Sender: TVPEngine
)

Sender
the VPE object that fired the event

Remarks:
If your application is not able to receive events, or if you don't want to process this event for some reason, you can test for AutoBreak (this means, check if an AutoBreak occurred after inserting a text object) with the following technique:

Store the current page-number in a variable. After the output, compare this variable to the current page-number. If it is different, an AutoBreak had occurred.

Code example:

```
var n := Doc.CurrentPage; // remember the current page number
Doc.Print(1, 1, '... very long text ...');
if n <> Doc.CurrentPage then
begin
  catched auto break!
  The formula "Doc.CurrentPage - n" returns the number of
  automatically generated pages.
end;
```

Notes:
If you are inserting text objects into the document while processing this event, make sure they will not cause again an Auto Page Break event - otherwise this will cause an endless recursion, ending up in a stack overflow with a GPF.

To avoid Auto Page Breaks, set AutoBreakMode = AUTO_BREAK_NO_LIMITS.

If you are modifying any properties, as for example the AutoBreakMode or FontSize etc., it will not be reset when you code exits the event handler. If you are changing properties, you need to save and restore them by code yourself. Do this by calling StoreSet() in the beginning of your event handler and by calling UseSet() followed by RemoveSet() at the end.

Example: if a Print() statement in your main code causes an AutoBreak and if your AutoBreak Handler sets the to something different than zero, then the rest of the text which is output by the previous Print() command of your main code will have a box around it.

Solution: at the very first beginning of your AutoBreak Handler call StoreSet() and on the last line call UseSet() and then RemoveSet(). Do this only if required, you can also check which single properties have been changed and change them back, because StoreSet() and UseSet() eat performance!

YOU MAY NOT USE NoHold of the embedded flags in the AutoBreak-Event Handler. This will cause an internal recursion and overwrite settings done in the main code. Explanation: When
using the flag NoHold in your main code in a `Print(Box)()` or `Write(Box)()` statement which causes an AutoBreak, the current settings are stored in a temporary memory block (to be restored after the command has been executed). Afterwards your AutoBreak Event-Handler is executed and if you use the NoHold Flag again, the temporary memory block is overwritten, so that its initial settings are lost.
5.11 OnRequestPrint Event - VCL

Is fired to inform the application about the several stages during the printing process.

OnRequestPrint(
    Sender: TVPEngine;
    Action: Integer;
    var ResultingAction: Integer
)

Sender
    the VPE object that fired the event

Action

<table>
<thead>
<tr>
<th>Action</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT_MSG_ABORT</td>
<td>0</td>
<td>User aborted while printing</td>
</tr>
<tr>
<td>PRINT_MSG_START</td>
<td>1</td>
<td>Print started</td>
</tr>
<tr>
<td>PRINT_MSG_END</td>
<td>2</td>
<td>Print ended</td>
</tr>
<tr>
<td>PRINT_MSG_SETUPABORT</td>
<td>3</td>
<td>User aborted Setup-Dialog</td>
</tr>
<tr>
<td>PRINT_MSG_SETUPSTART</td>
<td>4</td>
<td>Setup-Dialog started</td>
</tr>
<tr>
<td>PRINT_MSG_SETUPEND</td>
<td>5</td>
<td>Setup-Dialog ended</td>
</tr>
</tbody>
</table>

ResultingAction
    is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Resulting Action</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT_ACTION_OK</td>
<td>0</td>
</tr>
<tr>
<td>PRINT_ACTION_ABORT</td>
<td>1</td>
</tr>
</tbody>
</table>

Remarks:
    Do not call CloseDoc() when processing this event. You would terminate a module that is working.

Your application should return PRINT_ACTION_OK (zero) if it processes this message, except for Action = PRINT_MSG_SETUPSTART, where your application may return in addition PRINT_ACTION_ABORT (= 1) to abort the print job. PRINT_MSG_SETUPSTART is sent, when the user clicked the print button in the preview (or pushed the corresponding key). You have the option to abort the job, for you can then create and print internally a new document which is completely different to the preview.

Another use for the PRINT_MSG_SETUPSTART message is, to pre-initialize the Device Control Properties at this stage, before the printer setup dialog will be shown to the user.
5.12 OnPrintNewPage Event

Is fired only while printing exactly before printing a new page. The event is useful to change the Device Control Properties on-the-fly during printing. You may change all properties, except the device itself.

OnPrintNewPage(
    Sender: TVPEngine;
    Page: Integer;
    var ResultingAction: Integer
)

Sender
  the VPE object that fired the event

Page
  current page number that will be printed

ResultingAction
  is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Resulting Action</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT_ACTION_OK</td>
<td>0</td>
</tr>
<tr>
<td>PRINT_ACTION_CHANGE</td>
<td>1</td>
</tr>
</tbody>
</table>

Remarks:
  Your application should return PRINT_ACTION_OK (zero) if it processes this message without changing a printing device's properties.

  If a Device Control Property was changed, your application must return PRINT_ACTION_CHANGE ( = 1).

  • If you change the Device Control Properties during the print job, you must reset them to the original values when the print job has finished (see OnRequestPrint Event - VCL; with Action = PRINT_MSG_ABORT or Action = PRINT_MSG_END).

  • Changing the properties (like DevPaperBin, DevOrientation, DevPaperFormat, etc.) during the print job doesn’t work with some (buggy) printer drivers, for example some fax drivers and the HP4 M Plus driver (the HP4 PS driver should work with the HP4 M Plus printer!).

  • Some properties and methods don’t work with some printer drivers. For example "DevTTOption" doesn’t work with our HP4 and HP5 printer drivers on WiW 3.11 and NT 3.51, but it works with both drivers on Win95. This is a driver problem.

  • Printer drivers are manufactured by vendors independent of IDEAL Software; we make no warranty, implied or otherwise, regarding these product's performance or reliability.

  • Win32s is not officially supported by VPE. The Device Control Properties do not work under Win32s.

Known Problems:

Changing any Device Control Properties during the print job disables Duplex Printing on PCL Printers
SYMPTOMS
When you change a Device Control Property during the print job, it appears to disable Duplex (double-sided) printing when the target printer is a (Hewlett Packard) PCL printer.

CAUSE
PCL printers treat a change in paper size as a new print job that requires the printer to be initialized. This causes the printer to eject any page that is currently in the printer. The PCL printer drivers for Windows assume that the page size has been changed when a Device Control Property is changed during the print job, unless the orientation of the page has changed.

RESOLUTION
To prevent having a page ejected when changing a Device Control Property during the print job, make sure that the function is called only between individual sheets of paper. Changing a Device Control Property before printing odd-numbered pages is sufficient for most applications that use duplex printing. However, some applications require that you change the page orientation on a page-by-page basis. In this case, you can change Device Control Properties between individual sheets of paper if the orientation has changed.

STATUS
This behavior is by design.

MORE INFORMATION
Note that when this problem occurs the print job continues and the sheets of paper are passed through the printer's duplexer, but the sheets are only printed on one side.

Because of the page size initialization requirement for PCL printers, Windows PCL drivers treat the change of Device Control Properties differently. These drivers allow only the orientation to change between the front and back pages of a sheet of paper. This means that the change of a Device Control Property will eject the page unless the orientation (and only the orientation) has changed from the previous page. Returning PRINT_ACTION_CHANGE although no Device Control Property has been changed causes the printer to eject the page.

Returning PRINT_ACTION_CHANGE although no Device Control Property has been changed is unnecessary. By doing so, you risk having a page ejected from the printer, which has a high probability of occurring.

See also:
OnPrintDevData Event
5.13 OnPrintDevData Event

Is sent only while printing, exactly before printing a new page and immediately after OnPrintNewPage() has been sent. The only use for this event is to call DevSendData() in response.

DevSendData() enables your application to send escape sequences to the printing device. So it is possible to select for example an output paper bin by code (or whatever other functionality is provided by the connected printer).

OnPrintDevData(
    Sender: TVPEngine;
    Page: Integer;
    var ResultingAction: Integer
)

Sender
    the VPE object that fired the event

Page
    current page number that will be printed

ResultingAction
    is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Resulting Action</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT_ACTION_OK</td>
<td>0</td>
</tr>
<tr>
<td>PRINT_ACTION_CHANGE</td>
<td>1</td>
</tr>
</tbody>
</table>

Remarks:
Your application should return PRINT_ACTION_OK (zero) if it processes this message without calling DevSendData().

If it called DevSendData(), your application must return PRINT_ACTION_CHANGE ( = 1).

See also:
OnPrintNewPage()
5.14 OnBeforeMail Event

Is fired, when the user clicked the eMail-button in the preview (or pushed the corresponding key). The e-mail was not sent yet, therefore your application has now the option to set receivers, attachments, etc. by code.

OnBeforeMail(
    Sender: TVPEngine;
    var Cancel: Boolean
)

Sender
the VPE object that fired the event

Cancel
is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>cancel operation i.e. do not mail the document</td>
</tr>
<tr>
<td>False</td>
<td>continue operation</td>
</tr>
</tbody>
</table>

If you set this parameter to True, the operation is cancelled, i.e. VPE will not mail the document. Cancelling the operation allows you to display your own mail dialog and/or to execute your own mailing code.

See also:
E-Mail Functions
5.15 OnAfterMail Event

Is sent, after the user clicked the eMail-button in the preview (or pushed the corresponding key) and the e-mail has already been sent.

OnAfterMail(
    Sender: TVPEngine;
    Result: Integer
)

Sender
the VPE object that fired the event

Result
the status of the e-mail action, one of the VERR_xyz error codes

See also:
E-Mail Functions
5.16 OnObjectClicked Event

[Professional Edition and above]

A clickable object with an assigned ObjectID has been clicked with the mouse. This is one of the few events where CloseDoc() may be called while processing it.

OnObjectClicked(
    Sender: TVPEngine;
    ObjectID: LongInt
)

Sender
the VPE object that fired the event

ObjectID
the Object ID of the object that was clicked by the user

See also:
Clickable objects
5.17 OnUDOPaint Event

[Professional Edition and above]

Is fired as a notification from a User Defined Object (UDO). The object needs to be painted to the output device.

See the description of the User Defined Objects for information on how to process this event.

\[
\text{OnUDOPaint}(\text{Sender: TVPEngine})
\]

\text{Sender}  
the VPE object that fired the event

\text{See also:}  
User Defined Objects
5.18 OnAfterControlEnter

[Interactive Edition and above]

A Control received the focus.
Normally, this event is sent if the user presses the Tab or Backtab key, or clicks with the mouse onto a control. This event can be used to re-format the content of a control.

OnAfterControlEnter(
    Sender: TVPEngine;
    Obj: TVPEObject
)

Sender
    the VPE object that fired the event

Obj
    the object, which fired the event

See also:
    Interactive Documents
5.19 OnRequestControlExit

[Interactive Edition and above]

The user wishes to remove the focus from the currently focused Control. In response to this event your application can evaluate the value of the Control and decide, whether the current value is valid and the Control may lose the focus or not.

OnRequestControlExit(
    Sender: TVPEngine;
    Obj: TVPEObject;
    var CanExit: Boolean
)

Sender
the VPE object that fired the event

Obj
the object, which fired the event

CanExit
is a return-parameter, i.e. you can assign it one of the following values to control the resulting action of VPE:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, the control may loose the focus</td>
</tr>
<tr>
<td>False</td>
<td>no, the control may not loose the focus</td>
</tr>
</tbody>
</table>

See also:
Interactive Documents
5.20 OnAfterControlExit

[Interactive Edition and above]

The currently focused Control lost the focus. Normally, this event is sent if the user presses the Tab or Backtab key, or clicks with the mouse onto another control. When receiving this message, it is possible for you to force the focus to be set explicitly to a specific control by calling `SetFocusControlByName()` or `SetFocus()`. It is also possible to enable and disable other controls of the current form while processing this event. In addition this event can be used to re-format the content of a control.

```pascal
OnAfterControlExit(
  Sender: TVPEngine;
  Obj: TVPEObject
)
```

*Sender*
The VPE object that fired the event

*Obj*
The object, which fired the event

See also:
- Interactive Documents
5.21 OnAfterControlChange

[Interactive Edition and above]

A Control changed its value, i.e. the content of a Control was edited by the user or the value of an associated Field was changed by code.

Evaluating this event means, that your application is informed about every single keystroke or mouse-click, which modifies a Control's content.

If you are working with Fields that are associated with controls - as recommended - your application should not take care of this event.

This is one of the few events where CloseDoc() may be called while processing the event.

The event is not fired, if you set the value of a Control by code.

OnAfterControlChange(
    Sender: TVPEngine;
    Obj: TVPEObject
)

*Sender*  
The VPE object that fired the event

*Obj*  
The object, which fired the event

**Remarks:**

If you change the value of a Control by code, the AfterControlChange event is not fired. But if the Control is associated with a Field, the event AfterFieldChange is fired.

Vice versa, if you change the value of a Field by code, the AfterFieldChange event is not fired, but any Controls associated with the Field will fire the event AfterControlChange.

**See also:**

Interactive Documents
5.22 OnAfterFieldChange

[Interactive Edition and above]

A Field (not a control!) changed its value, because the associated Control was edited by the user or the content of any associated Control was changed by code. This event is very interesting, because a Field will change its value each time the user makes a change. Evaluating this event means, that your application is informed about every single keystroke or mouse-click, which modifies a Control's content. This is one of the few events where CloseDoc() may be called while processing the event. The event is not fired, if you set the value of a Field by code.

OnAfterFieldChange(
    Sender: TVPEngine;
    Obj: TVPEField
)

Sender
the VPE object that fired the event

Obj
the Field object, which fired the event

Remarks:
If you change the value of a Control by code, the OnAfterControlChange event is not fired. But if the Control is associated with a Field, the event AfterFieldChange is fired. Vice versa, if you change the value of a Field by code, the AfterFieldChange event is not fired, but any Controls associated with the Field will fire the event OnAfterControlChange.

See also:
Interactive Documents
6 Management

6.1 Management
Management properties and methods deal with the control of VPE itself. With the creation of virtual documents, storing and retrieving them from/to file, handling the preview, etc.
6.2 OpenDoc

Creates a new document with one initial blank page, reflecting the current state of all Design Time Properties. You always need to open the document before you may access any methods or runtime properties.

**method void VPE.OpenDoc( )**

**Remarks:**

In case of an Error, **LastError** is set.

You may create an unlimited number of pages per document and an unlimited number of documents simultaneously, but both is limited by available memory. How much memory is needed, depends on the number of objects you insert and what type of objects you insert (a bitmap for example needs much more memory than a single line). So we can't give clear guidelines about memory usage. In case of doubt, use a monitoring tool to view how much memory is needed for your specific kind of document(s). For example, one page in the "Speed + Tables" demo needs about 10 KB of memory. This is really low, but 100 pages need about 1 MB of memory. If the memory usage is too high, we recommend to use **File Swapping**.

A VPE document can exist without showing a preview. But if a preview is shown, the document is closed and removed from memory by default, when the preview is closed by the user or when the parent window is closed. If you set **AutoDelete** = false, the document is not closed when the preview is closed.

On non-Windows platforms you can call **CloseDoc()** to remove a document from memory, or destroy the component itself.

To prevent the runtime error of opening a document that is already open, you have two possible options:

1. You check the property "**IsOpen**":

   ```vba
   Private Sub ButtonReport_Click()
   If Report.IsOpen Then Exit Sub
   Call Report.OpenDoc
   Call Report.VpePrint(1, 1, "Hello World!")
   Call Report.Preview
   End Sub
   ```

   or,

2. You disable the control (button, menu-entry or whatsoever), which will cause opening the document, until the document is closed:

   ```vba
   Private Sub ButtonReport_Click()
   ButtonReport.Enabled = False
   Call Report.OpenDoc
   Call Report.VpePrint(1, 1, "Hello World!")
   Call Report.Preview
   End Sub
   ```

The closing of the document fires the event "**DestroyWindow**", so we enable the button here again:

```vba
Private Sub Report_DestroyWindow()
   ButtonReport.Enabled = True
End Sub
```
6.3 IsOpen

Returns the status of the document.

property boolean VPE.IsOpen

read; runtime only

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>the document is open (OpenDoc has been called)</td>
</tr>
<tr>
<td>False</td>
<td>the document is not open</td>
</tr>
</tbody>
</table>

Remarks:

This property does not return the status of the Preview. To check if the Preview is open use the property IsPreviewVisible.
6.4 CloseDoc

Closes the specified document (and also the preview, if open).

**method int VPE.CloseDoc( )**

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Ok</td>
</tr>
<tr>
<td>False</td>
<td>couldn’t close, because the document is currently printed</td>
</tr>
</tbody>
</table>

**Remarks:**

You may call this method even if the document is already closed. In this case the method will return True.

**Note:** when a form which contains the VPE Control is being closed, it is very important that you call CloseDoc. You need to call CloseDoc exactly at the time when receiving the event which asks for confirmation if the form can be closed. (For C# and Visual Basic .NET it is the **Form_Closing()** event, for Visual Basic it is the **Form_QueryUnload()** event, for Delphi and C++ Builder it is the **TForm.OnClose()** event.)

In case CloseDoc should return False in that moment, you need to cancel the process of closing the window, i.e. the form must not be closed.

**Example for Visual Basic .NET:**

```vbnet
Private Sub MenuClose_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MenuClose.Click
' Do NOT call Application.Exit(), because Form1_Closing() will not be executed then!
Close()
End Sub

Private Sub Form1_Closing(ByVal sender As Object, ByVal e As System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
If Not Usage.CloseDoc Or Not Enhanced.CloseDoc Then
    MsgBox("The application cannot terminate until all jobs have finished printing.")
e.Cancel = True
End If
End Sub
```
Example for C#:

```csharp
private void menuClose_Click(object sender, System.EventArgs e)
{
    // Do NOT call Application.Exit(), because Form1_Closing() will not be
    // executed then!
    Close();
}

private void Form1_Closing(object sender,
    System.ComponentModel.CancelEventArgs e)
{
    {
        MessageBox.Show("The application cannot terminate until all jobs have
                        finished printing.");
        e.Cancel = true;
    }
}
```

Example for Visual Basic:

```vbnet
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    If Not VPE1.CloseDoc Or Not VPE2.CloseDoc Then
        MsgBox ("The application cannot terminate until all jobs have
                finished printing.")
        Cancel = True
    End If
End Sub
```

Example for Delphi:

```delphi
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    if not VPE1.CloseDoc or not VPE2.CloseDoc then
    begin
        ShowMessage('The application cannot terminate until all jobs have
                     finished printing.');
        Action := caNone;
    end;
end;
```

In the above examples the form contains two VPE Controls named "VPE1" and "VPE2". For both controls it is checked whether their previews can be closed or not.
6.5 SwapFileName

If this property is set, VPE will work with a Swap File when `OpenDoc()` is called.

Instead of storing all document pages in memory, only the current page is held in memory. All other pages are swapped to a VPE document file. This implies minimum memory usage at very high performance and allows to create huge documents. VPE's file swapping is VERY fast.

Even for file-based documents you can add new pages to the end of a document at any point in time.

Editions below the Professional Edition: after a page has been swapped to file, you cannot modify the page, i.e. add new objects to it.

The Professional Edition and higher allow to add new objects to pages which have already been written to file and to clear, insert and delete pages at any position in a document file.

A page is swapped to file after:

- Adding a new blank page by calling `PageBreak()`
- Moving to a different page by modifying the property `CurrentPage`

For details about creating and using VPE document files, please see the "Programmer's Manual", chapter "Programming Techniques", subchapter "VPE Document Files".

**property string VPE.SwapFileName**

read / write; design & runtime, closed document required

**Possible Values:**

swap file name (= document file name)

**Default:**

empty = not set

**Remarks:**

it is very important, that all VPE document files have the suffix ".vpe". Always use this suffix, because "VPE View" (the document browser) is associated with this suffix.

You can check for error conditions - for example, if there is not enough free space left on disk for the SwapFile - by testing the property `LastError` after calling `OpenDoc` and each time after calling `PageBreak`.

When creating a new document with a swap file, `compression` is always activated.

**Example:**

```vpe
Doc.SwapFileName = "c:\reports\report.vpe"
Doc.OpenDoc
```

Instructs VPE to use the file "c:\reports\report.vpe" as Swap File.

```vpe
Doc.SwapFileName = ""
```

Instructs VPE to use no Swap File (the default).
Example:

```vpe
long count
Doc.SwapFileName = "c:\docs\report1.vpe"
Doc.OpenDoc
count = Doc.PageCount
Doc.PageBreak
Doc.Print(1, 1, "Added a new page.")
Doc.Preview
```

If the VPE document file "c:\docs\report1.vpe" is already existing, the file will be opened and the first page is read into memory. If the document file is not existing, VPE will create it with an initial blank page. The variable "count" is assigned the number of pages the document contains. VPE will add a new page at the end of the document and insert the text "Added a new page." there. Then the preview is shown.

The property **DocFileReadOnly**:

```vpe
long count;
Doc.SwapFileName = "c:\docs\report1.vpe"
Doc.DocFileReadOnly = True
Doc.OpenDoc
if Doc.LastError <> 0 then
  exit function
end if
count = Doc.PageCount
Doc.VisualPage = count
Doc.Preview
```

If the VPE document file "c:\docs\report1.vpe" is not existing, the property **LastError** will return VERR_FILE_OPEN. Otherwise the file will be opened in read-only mode and the first page is read into memory. If the document file is not existing, VPE will create it with an initial blank page. The variable "count" is assigned the number of pages the document contains. The preview will show the last page contained in the document.

When using the SwapFileName property, **DevJobName** is set automatically to the file name.

See Also:  
**WriteDoc** and **ReadDoc**
6.6 EditProtection

[Professional Edition and above]

The Visual Designer dycodoc can read and edit VPE Document files which have been created with the VPE Professional Edition or any higher edition.

If you want to protect your VPE Document files so they can not be read by dycodoc, call this function. In both cases - either if a document was opened using SwapFileName or if you call WriteDoc() - the created VPE document file will be protected.

Interactive Edition only: if the EditProtection is enabled, the interactive objects stored in VPE Document files are not editable within VPE, nor VPEView.

Once you have called this function, it is impossible to unprotect the document.

property EditProtection [integer] VPE.EditProtection

read / write; runtime only

**ActiveX / VCL:**
You must assign the value "1" to this property! Different values are reserved for future extensions. Do not assign any different value!

**.NET:**
possible values are:
enum EditProtection
{
    Unprotected,
    Protected
}

**Default:**
Unprotected (0) = the current document is not protected from being edited with dycodoc

**Remarks:**
Once you have assigned a value to this property, it is impossible to unprotect the document. For security reasons, dycodoc can not read VPE Document files which have been created with any release prior to VPE v3.20.

**Example:**

**ActiveX / VCL:**
VPE.EditProtection = 1
VPE.WriteDoc("my_file.vpe")

**.NET:**
VPE.EditProtection = EditProtection.Protected
VPE.WriteDoc("my_file.vpe")

Activates the edit protection and writes the current document as protected file to disk.
6.7 License

[Java, PHP, Python, Ruby, VCL Only]

Licences VPE or an add-on module for the given document, so the demo banners disappear.

**method void VPE.License**
```java
    string Serial1,
    string Serial2
```

*string Serial1, Serial2*  
the two serial strings provided by IDEAL Software when you acquire a license

**Example:**
If the license key has the following form:
VPE-A1234-123456
ABCD-EFGH

The method is called with:

```java
VPE.License("VPE-A1234-123456", "ABCD-EFGH")
```

**Remarks:**
If you are using the VPE ActiveX, or the .NET component, this method is not available. The licensing for the ActiveX is done automatically by the COM Licensing Mechanism as defined by Microsoft. For a detailed explanation of the COM Licensing Mechanism, please see "Installing the VPE ActiveX - The Demo Banners Are Still Shown" in the Programmer's Manual.

If you obtained additional license keys for add-on products, call this method for each License Key separately. It is necessary that the VPE module is licensed first before any add-on module is licensed. Otherwise the licensing of the add-on modules will fail.

If you are using a Server License, it is necessary that you call License() for all available add-on modules first, before setting the property `EnableMultiThreading` = `true`. 
6.8 EnableMultiThreading

[Professional Edition or higher]

By default, VPE is not thread-safe. If you are using VPE in a multi-threaded environment, set the property `EnableMultiThreading = True`, to activate the thread-safe code of VPE.

On some platforms it might be required to purchase and install a special server license (for each server), before this property can be used.

**property boolean VPE.EnableMultiThreading**

- write; runtime

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>VPE’s thread-safe code is activated</td>
</tr>
<tr>
<td>False</td>
<td>VPE’s thread-safe code is not active</td>
</tr>
</tbody>
</table>

**Default:**

False

**Remarks:**

VPE documents must be created and used separately per thread, i.e. each thread must create a VPE document itself by calling `OpenDoc()`, and one thread must not use the document handle of another thread for calls to the VPE API.

VPE will operate slower, if the thread-safe code is activated, due to acquiring and releasing thread-locks.

Once the thread-safe code has been activated, it is activated for all VPE documents that are currently open - and that will be opened later - by the calling application instance. Furthermore the thread-safe code can not be deactivated by the application instance.

The trial versions of VPE allow to activate the thread-safe code for testing purposes.

The following does not apply to the .NET and ActiveX components, since both perform the licensing internally:

If you activated multi-threading and call later the `License()` method, the licensing will fail. You must call the License() method **before** you activate the thread-safe code.
6.9 EnableURLs

[ActiveX only]

Instructs the ActiveX to redirect all file accesses to URL’s, e.g. via the internet. This enables you to load documents, images, and Rich Text (RTF) from servers and internet sites. It is therefore ideal to use the ActiveX with VBScript or JavaScript within an Internet Browser. Even images INSIDE of a VPE document file - that reference a URL - are loaded via the network.

**property boolean VPE.EnableURLs**

read / write; design & runtime, closed document required

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Enabled</td>
</tr>
<tr>
<td>False</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

**Default:**

False (do not interpret file names as URL’s)

**Example:**

```vba
doc.EnableURLs = True
Doc.Picture(1, 1, VFREE, VFREE, "ftp://ftp.my-server.com/image1.gif")
```

If the VPE ActiveX is embedded within a Browser-HTML page, you can also use relative paths. For example if the HTML file with the embedded VPE ActiveX was loaded from:

"http://www.my-server.com/usa/reports/active.html"

and the embedded VPE ActiveX shall load the image:

"http://www.my-server.com/usa/reports/images/img1.jpg"

the following call will do when creating the document itself:

```vba
VpeControl.Picture(1, 1, VFREE, VFREE, "images/img1.jpg")
```

**To read a file from the local hard drive with EnableURLs = True, use the following:**

```vba
Doc.ReadDoc("file://c:\docs\report.vpe")
Doc.Picture(1, 1, VFREE, VFREE, "file://c:\pictures\img1.jpg")
```

In the directory "Internet" - located in the VPE installation directory - is a complete HTML source code example. It demonstrates the use of VPE embedded within a Browser and JavaScript. The HTML source code example contains many helpful comments, which should be studied carefully.
6.10 LastError

Returns the error state of the last VPE function call.

**property ErrorCode [long] VPE.LastError**

read; runtime only

**Possible Values:**
one of the VERR_xyz constants (see below)

**Remarks:**
Not all functions do set / clear the error state. Only the functions which set the error state will also clear it, in case that no error occurred. All other functions keep the error state untouched.

The "Remarks" section of each function described in this manual will clearly indicate, if the function will modify the LastError property.

**Error Codes:**

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Value</th>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERR_OK</td>
<td>0</td>
<td>Ok</td>
<td>no error</td>
</tr>
<tr>
<td>VERR_COMMON</td>
<td>1</td>
<td>Common</td>
<td>common error</td>
</tr>
<tr>
<td>VERR_CANCELLLED</td>
<td>2</td>
<td>Cancelled</td>
<td>the user cancelled an operation, for example the Open or Save file dialog</td>
</tr>
<tr>
<td>VERR_MEMORY</td>
<td>100</td>
<td>Memory</td>
<td>out of memory</td>
</tr>
<tr>
<td>VERR_FILE_OPEN</td>
<td>200</td>
<td>FileOpen</td>
<td>error opening a file; occurs when calling functions like OpenDoc() with</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SwapFileName set, ReadDoc(), WriteDoc(), ReadPrinterSetup(), Picture(),</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WriteRTFFile(), etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Meaning: The specified file could not be opened, it is either not existing,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the path is not existing, it is locked, incorrect logical structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(for example corrupted file)</td>
</tr>
<tr>
<td>VERR_FILE_DOCVERSION</td>
<td>201</td>
<td>FileDocVersion</td>
<td>Document file has the wrong (higher) version and can not be opened / read</td>
</tr>
<tr>
<td>VERR_FILE_CREATE</td>
<td>202</td>
<td>FileCreate</td>
<td>error creating file; occurs when calling functions like WriteDoc(), OpenDoc()</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>with SwapFileName set, WritePrinterSetup(), SetupPrinter() (only during</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>write), etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Meaning: illegal path or file name, file locked, disk full, no permissions</td>
</tr>
<tr>
<td>VERR_FILE_ACCESS</td>
<td>203</td>
<td>FileAccess</td>
<td>Access denied (no permission); see also</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DocFileReadOnly</td>
</tr>
<tr>
<td>Error Code</td>
<td>Description</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>VERR_FILE_READ</td>
<td>204</td>
<td>FileRead</td>
<td>error during file read operation</td>
</tr>
<tr>
<td>VERR_FILE_WRITE</td>
<td>205</td>
<td>FileWrite</td>
<td>error during file write operation</td>
</tr>
<tr>
<td>VERR_PRINT_SETUP_ABORT</td>
<td>225</td>
<td>PrintSetupAbort</td>
<td>Printer setup was aborted by user</td>
</tr>
<tr>
<td>VERR_PRINT_SETUP_INIT</td>
<td>226</td>
<td>PrintSetupInit</td>
<td>Printer setup failed: initialization of printer</td>
</tr>
<tr>
<td>VERR_PRINT_SYS</td>
<td>227</td>
<td>PrintSys</td>
<td>Failure in the Windows printing subsystem</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Possible causes might include a network printer that has been renamed,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>corrupt printer drivers, or corrupt print subsystem files.</td>
</tr>
</tbody>
</table>

**Solution or Workaround**

In an effort to solve the problem, use the steps below to remove and reinstall the printer. If the problem persists, it may suggest that there are further problems with the operating system or printer itself. Contact your system administrator for further assistance.

- Go to Start > Settings > Printers. Remove the printer by selecting it and choose Delete from the File menu.
- Choose File > Server Properties to open up the Server Properties dialog.
- Click on the Drivers tab.
- Find the printer's name in the list and remove it. There may be multiple entries for the same printer; be sure to remove all instances of the printer driver.
- Reinstall the printer normally by returning to the Start > Settings > Printers and clicking on click the Add Printer icon.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERR_PRINT_COMMON</td>
<td>228</td>
<td>PrintCommon</td>
</tr>
<tr>
<td>VERR_PIC_IMPORT</td>
<td>300</td>
<td>PicImport</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error Code</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VERR_PIC_NOLICENSE</td>
<td>301</td>
<td>PicNoLicense No license for image access (e.g. TIFF / GIF image is LZW compressed and flag PIC_ALLOWLZW not used). This error code is obsolete. The LZW patent has expired. Since v4.00 VPE imports LZW compressed images without specifying PIC_ALLOWLZW.</td>
</tr>
<tr>
<td>VERR_PIC_DXFCoord</td>
<td>302</td>
<td>PicDXFCoord For DXF formats, x2 and y2 may not be VFREE at the same time, either x2 or y2 must be &lt;&gt; VFREE</td>
</tr>
<tr>
<td>VERR_PIC_EXPORT</td>
<td>350</td>
<td>PicExport Image could not be exported Meaning: File could not be created, or not enough memory</td>
</tr>
<tr>
<td>VERR_MOD_GRAPH_IMP</td>
<td>400</td>
<td>ModGraphImp Error loading Graphics Import Library</td>
</tr>
<tr>
<td>VERR_MOD_GRAPH_PROC</td>
<td>401</td>
<td>ModGraphProc Error loading Graphics Processing Library</td>
</tr>
<tr>
<td>VERR_MOD_BARCODE</td>
<td>402</td>
<td>ModBarcode Error loading Barcode Library</td>
</tr>
<tr>
<td>VERR_MOD_CHART</td>
<td>403</td>
<td>ModChart Error loading Chart Library</td>
</tr>
<tr>
<td>VERR_MOD_ZLIB</td>
<td>404</td>
<td>ModZlib Error loading ZLIB Library</td>
</tr>
<tr>
<td>VERR_MOD_VPDF</td>
<td>405</td>
<td>ModPDF Error loading PDF Export Library</td>
</tr>
<tr>
<td>VERR_MOD_VBAR2D</td>
<td>406</td>
<td>ModBarcode2D Error loading 2D Barcode Library</td>
</tr>
<tr>
<td>VERR_MAIL_LOAD_MAPI</td>
<td>450</td>
<td>MailLoadMapi Could not load MAPI</td>
</tr>
<tr>
<td>VERR/mail_CREATE</td>
<td>451</td>
<td>MailCreate Could not create temporary file</td>
</tr>
<tr>
<td>VERR_MAIL_USER_ABORT</td>
<td>452</td>
<td>MailUserAbort</td>
</tr>
<tr>
<td>VERR_MAIL_FAILURE</td>
<td>453</td>
<td>MailFailure</td>
</tr>
<tr>
<td>VERR_MAIL_LOGON_FAILURE</td>
<td>454</td>
<td>MailLogonFailure</td>
</tr>
<tr>
<td>VERR_MAIL_DISK_FULL</td>
<td>455</td>
<td>MailDiskFull</td>
</tr>
<tr>
<td>VERR_MAIL_INSUFFICIENT_MEMORY</td>
<td>456</td>
<td>MailInsufficientMemory</td>
</tr>
<tr>
<td>VERR_MAIL_ACCESS_DENIED</td>
<td>457</td>
<td>MailAccessDenied</td>
</tr>
<tr>
<td>VERR_MAIL_RESERVED</td>
<td>458</td>
<td>MailReserved</td>
</tr>
<tr>
<td>VERR_MAIL_TOO_MANY_SESSIONS</td>
<td>459</td>
<td>MailTooManySessions</td>
</tr>
<tr>
<td>VERR_MAIL_TOO_MANY_FILES</td>
<td>460</td>
<td>MailTooManyFiles</td>
</tr>
<tr>
<td>VERR_MAIL_TOO_MANY_RECIPIENTS</td>
<td>461</td>
<td>MailTooManyRecipients</td>
</tr>
<tr>
<td>VERR_MAIL_ATTACHMENT_NOT_FOUND</td>
<td>462</td>
<td>MailAttachmentNotFound</td>
</tr>
<tr>
<td>VERR_MAIL_ATTACHMENT_OPEN_FAILURE</td>
<td>463</td>
<td>MailAttachmentOpenFailure</td>
</tr>
<tr>
<td>VERR_MAIL_ATTACHMENT_WRITE_FAILURE</td>
<td>464</td>
<td>MailAttachmentWriteFailure</td>
</tr>
<tr>
<td>VERR_MAIL_UNKNOWN_RECIPIENT</td>
<td>465</td>
<td>MailUnknownRecipient</td>
</tr>
<tr>
<td>VERR_MAIL_BAD_RECYPTYPE</td>
<td>466</td>
<td>MailBadRecipType</td>
</tr>
<tr>
<td>VERR_MAIL_NO_MESSAGES</td>
<td>467</td>
<td>MailNoMessages</td>
</tr>
<tr>
<td>VERR_MAIL_INVALID_MESSAGE</td>
<td>468</td>
<td>MailInvalidMessage</td>
</tr>
<tr>
<td>VERR_MAIL_TEXT_TOO_LARGE</td>
<td>469</td>
<td>MailTextTooLarge</td>
</tr>
<tr>
<td>Error Code</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------</td>
<td></td>
</tr>
<tr>
<td>VERR_MAIL_INVALID_SESSION 470</td>
<td>MailInvalidSession</td>
<td></td>
</tr>
<tr>
<td>VERR_MAIL_TYPE_NOT_SUPPORTED 471</td>
<td>MailTypeNotSupported</td>
<td></td>
</tr>
<tr>
<td>VERR_MAIL_AMBIGUOUS_RECIPIENT 472</td>
<td>MailAmbiguousRecipient</td>
<td></td>
</tr>
<tr>
<td>VERR_MAIL_MESSAGE_IN_USE 473</td>
<td>MailMessageInUse</td>
<td></td>
</tr>
<tr>
<td>VERR_MAIL_NETWORK_FAILURE 474</td>
<td>MailNetworkFailure</td>
<td></td>
</tr>
<tr>
<td>VERR_MAIL_INVALID_EDITFIELDS 475</td>
<td>MailInvalidEditFields</td>
<td></td>
</tr>
<tr>
<td>VERR_MAIL_INVALID_RECIPS 476</td>
<td>MailInvalidRecips</td>
<td></td>
</tr>
<tr>
<td>VERR_MAIL_NOT_SUPPORTED 477</td>
<td>MailNotSupported</td>
<td></td>
</tr>
<tr>
<td>VERR_ZLIB_STREAM 500</td>
<td>ZlibStream Stream Inconsistent</td>
<td></td>
</tr>
<tr>
<td>VERR_ZLIB_DATA 501</td>
<td>ZlibData Data Corrupt</td>
<td></td>
</tr>
<tr>
<td>VERR_ZLIB_BUFFER 502</td>
<td>ZlibBuffer Internal Buffer Error</td>
<td></td>
</tr>
<tr>
<td>VERR_ZLIB_VERSION 503</td>
<td>ZlibVersion Wrong Version of ZLIB Library</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_FORMAT_OUT_OF_RANGE 550</td>
<td>Vbar2DFormatOutOfRange</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_UNDEFINED_ID 551</td>
<td>Vbar2DUndefinedId</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_FORMAT_TOO_LONG 552</td>
<td>Vbar2DFormatTooLong</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_FORMAT_OUT_OF_MEMORY 553</td>
<td>Vbar2DFormatOutOfMemory</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_FORMAT_DATA_INVALID 554</td>
<td>Vbar2DFormatDataInvalid</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_FORMAT_NOT_ALLOWED 555</td>
<td>Vbar2DFormatNotAllowed</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_DATA_WRONG_LENGTH 556</td>
<td>Vbar2DDataWrongLength</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_DATA_ZERO_LENGTH 557</td>
<td>Vbar2DDataZeroLength</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_DATA_TOO_SHORT 558</td>
<td>Vbar2DDataTooShort</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_DATA_TOO_LONG 559</td>
<td>Vbar2DDataTooLong</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_INVALID_DATA 560</td>
<td>Vbar2DInvalidData</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_SQUARE_EDGE_TOO_SMALL 561</td>
<td>Vbar2DSquareEdgeTooSmall</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_SQUARE_TOO_LARGE 562</td>
<td>Vbar2DSquareTooLarge</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_EDGE_OVER_FORCED 563</td>
<td>Vbar2DEdgeOverForced</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_SQUARE_ASPECT_SMALL 564</td>
<td>Vbar2DSquareAspectSmall</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_SQUARE_ASPECT_LARGE 565</td>
<td>Vbar2DSquareAspectLarge</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_SQUARE_EVEN_ODD_MATCH 566</td>
<td>Vbar2DSquareEvenOddMatch</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_INVALID_EDGE 567</td>
<td>Vbar2DInvalidEdge</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_SQUARE_EDGE_TOO_LARGE 568</td>
<td>Vbar2DSquareEdgeTooLarge</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_INVALID_ECC 569</td>
<td>Vbar2DInvalidEcc</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_INVALID_BORDER 570</td>
<td>Vbar2DInvalidBorder</td>
<td></td>
</tr>
<tr>
<td>VERR_VBAR2D_SELF_TESTFAILED 571</td>
<td>Vbar2DSelfTestFailed</td>
<td></td>
</tr>
<tr>
<td>VERR_RTF_BRACES 1000</td>
<td>RtfBraces RTF: unbalanced braces &quot;[]&quot;</td>
<td></td>
</tr>
<tr>
<td>VERR_RTF_OVERFLOW 1001</td>
<td>RtfOverflow RTF: only 16-bit version; generated internal structure &gt; 64 KB</td>
<td></td>
</tr>
<tr>
<td>VERR_RTF_FONTTBL 1002</td>
<td>RtfFontTable RTF: error parsing font table</td>
<td></td>
</tr>
<tr>
<td>VERR_RTF_COLORTBL 1003</td>
<td>RtfColorTable RTF: error parsing color table</td>
<td></td>
</tr>
<tr>
<td>VERR_TPL_OWNERSHIP 2000</td>
<td>TplOwnership Template: tried to dump the template to a foreign VPE document (where the template was not loaded into).</td>
<td></td>
</tr>
<tr>
<td>Error Code</td>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>VERR_TPL_PAGE_ALREADY_DUMPED</td>
<td>2001</td>
<td>TplPageAlreadyDumped</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Template: the page contains interactive objects and already had been dumped. A page containing interactive objects may only be dumped once.</td>
</tr>
<tr>
<td>VERR_TPL_AUTHENTICATION</td>
<td>2002</td>
<td>TplAuthentication</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Template: the template has an authentication key and it has not been validated successfully</td>
</tr>
</tbody>
</table>
6.11 Preview

[GUI Control Only]

Opens the preview window. The preview shows the page specified by the property VisualPage (by default, this is the first page). You can embed the preview into your own window / form, see the property External Window.

method void VPE.Preview(
)

See also:

  PreviewDoc
6.12 PreviewDoc

[GUI Control Only]

The same as Preview(). Opens the preview window on a defined position. The preview shows the page specified by VisualPage (by default, this is the first page). You can embed the preview into your own window / form, see the property External Window.

```java
method void VPE.PreviewDoc(
    long Left,
    long Top,
    long Right,
    long Bottom,
    integer ShowHide
)
```

*long Left, Top, Right, Bottom*
position of the preview window in pixels, for "Left" = -1 all coordinates are ignored.

*integer ShowHide*
can be one of the following predefined constants:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPE_SHOW_NORMAL</td>
<td>1: show window normal</td>
</tr>
<tr>
<td>VPE_SHOW_MAXIMIZED</td>
<td>2: show window maximized</td>
</tr>
<tr>
<td>VPE_SHOW_HIDE</td>
<td>3: hide the preview window</td>
</tr>
</tbody>
</table>
6.13 CenterPreview

[GUI Control Only]
The preview-window is centered in the middle of the parent window (= parent form) or desktop.

**method void VPE.CenterPreview(**
  
  long Width,
  
  long Height,
  
  boolean OnScreen
  
  )

*long Width, Height*

the width and height of the preview window in pixels, they can take the following special values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width 1</td>
<td>the preview's width is kept</td>
</tr>
<tr>
<td>Height 1</td>
<td>the preview's height is kept</td>
</tr>
<tr>
<td>Width 2</td>
<td>the preview's width is set to the width of the parent window (or desktop)</td>
</tr>
<tr>
<td>Height 2</td>
<td>the preview's height is set to the height of the parent window (or desktop)</td>
</tr>
</tbody>
</table>

*boolean OnScreen*

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>the preview is centered on the desktop (screen)</td>
</tr>
<tr>
<td>False</td>
<td>the preview is centered relative to the parent window</td>
</tr>
</tbody>
</table>

**Remarks:**

Java: The parameter `OnScreen` has no effect. The preview is always centered on the desktop.
6.14 BringPreviewToTop

[GUI Control Only]

Brings the VPE preview-window to the foreground and sets the focus to it. This only works, if the preview window is not embedded, and if another window of the calling process has currently the focus.

```java
method void VPE.BringPreviewToTop()
```
6.15 PreviewCtrl

[GUI Control Only]

Defines the behavior of the preview for page forward / backward actions. It specifies the vertical positioning of the page in the preview after the user moved a page forward / backward.

**property PreviewCtrl [integer] VPE.PreviewCtrl**

*read / write; design & runtime (read access is only possible, if the document is closed)*

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREVIEW_STAY</td>
<td>0</td>
<td>Stay</td>
<td>vertical position unaffected</td>
</tr>
<tr>
<td>PREVIEW_JUMPTOP</td>
<td>1</td>
<td>JumpTop</td>
<td>vertical position on top of page (default)</td>
</tr>
</tbody>
</table>

**Default:**

PREVIEW_JUMPTOP

**Example:**

**ActiveX / VCL:**

```csharp
doc.PreviewCtrl = PREVIEW_STAY
```

**.NET:**

```csharp
doc.PreviewCtrl = PreviewCtrl.Stay
```
6.16 ClosePreview

[GUI Control Only]

Closes the preview. The document is NOT closed (= removed from memory), regardless of the setting of AutoDelete. So you can re-open the preview with the method Preview() after a call to this method.

```java
method void VPE.ClosePreview()
```

6.17 IsPreviewVisible

[GUI Control Only]

Returns the visibility state of the Preview.

**property boolean VPE.IsPreviewVisible**

read; runtime only

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>the Preview is visible</td>
</tr>
<tr>
<td>False</td>
<td>the Preview is invisible (hidden)</td>
</tr>
</tbody>
</table>
6.18 SetFocus

[ActiveX / VCL only – see also the method Focus]

If the Preview is not embedded, a call to this method sets the focus to the Preview.

```csharp
method void VPE.SetFocus(
)
```

Remarks:

For Visual Basic use the method `VpeSetFocus`, because Visual Basic does not understand that a method of an ActiveX control is called and mismatches it with a call to the VB internal method.
6.19 VpeSetFocus

[ActiveX only]

If the Preview is not embedded, a call to this method sets the focus to the Preview.

method void VPE.VpeSetFocus(
)
6.20 Focus

[GUI Control Only, .NET and Java only]

Sets input focus to the control. If the Preview is not embedded, the external preview window will receive the focus.

method bool VPE.Focus( )

Returns:
true if the input focus request was successful; otherwise false.
6.21 VisualPage

[GUI Control Only]

Retrieve or set the number of the currently visible page in the preview. This page is independently from the currently active page, where you can insert objects on (see CurrentPage).

If you set this property, the preview display the specified page. VPE has internally two working pages: one your application is working on and one - the visual page - which is currently viewed by the user (if the preview is open). With this property you can set / retrieve the position of the visual page.

property long VPE.VisualPage

    read / write; runtime only

Possible Values:
The number of the currently visible page.

Default:
1 = the first page, after a call to OpenDoc()

Remarks:
In case of an error, LastError is set.

If want to show the preview and let the user work with it (e.g. scroll and print) while your application is still generating the document, see "Multipage Documents" in the Programmer's Manual for details.

Example:

// retrieve the page that is currently shown in the preview:
n = Doc.VisualPage

// move the preview to the next page:
Doc.VisualPage = n + 1
6.22 DispatchAllMessages

[GUI Control Only]

Needs to be called regularly to allow the user to browse in the preview through an open document while your application is still generating the report (i.e. adding objects and pages).


method boolean VPE.DispatchAllMessages(
)

Returns:

The method returns True, if the preview - or the parent window of the preview - are closed by the user. In such a case you should perform the necessary cleanup (for example close tables and databases) and exit the function that creates the report immediately. You may not call any VPE function of the related document, if this function returns True.

The method returns False, if the preview or the parent window of the preview are NOT closed by the user.

Remarks:

Never call DispatchAllMessages() while processing a VPE Event!
6.23 Refresh

[GUI Control Only]

When the preview is open, this call will refresh the preview and make all changes to the current page visible. If the user scrolls a page, all changes to the document will be visible automatically.

method void VPE.Refresh(
)

6.24 DocExportType

The value of this property controls, what type of document is created by the method WriteDoc().

**property DocExportType [integer] VPE.DocExportType**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPE_DOC_TYPE_AUTO</td>
<td>0</td>
<td>Auto</td>
<td>WriteDoc() will create VPE, PDF, HTML, XML or ODT files, depending on the file suffix of the supplied file name</td>
</tr>
<tr>
<td>VPE_DOC_TYPE_VPE</td>
<td>1</td>
<td>VPE</td>
<td>WriteDoc() will create VPE files, regardless of the file suffix</td>
</tr>
<tr>
<td>VPE_DOC_TYPE_PDF</td>
<td>2</td>
<td>PDF</td>
<td>WriteDoc() will create PDF files, regardless of the file suffix</td>
</tr>
<tr>
<td>VPE_DOC_TYPE_HTML</td>
<td>3</td>
<td>HTML</td>
<td>WriteDoc() will create HTML files, regardless of the file suffix</td>
</tr>
<tr>
<td>VPE_DOC_TYPE_XML</td>
<td>4</td>
<td>XML</td>
<td>WriteDoc() will create XML files, regardless of the file suffix</td>
</tr>
<tr>
<td>VPE_DOC_TYPE_ODT</td>
<td>5</td>
<td>ODT</td>
<td>WriteDoc() will create ODT files, regardless of the file suffix</td>
</tr>
</tbody>
</table>

**Default:**

VPE_DOC_TYPE_AUTO
6.25 Compression
[Not supported by the Community Edition]
Returns / sets the current mode for file compression. This affects writing native VPE document files as well as exporting PDF files.

property Compression [long] VPE.Compression

    read / write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOC_COMPRESS_NONE</td>
<td>0</td>
<td>None</td>
<td>no compression</td>
</tr>
<tr>
<td>DOC_COMPRESS_FLATE</td>
<td>1</td>
<td>Flate</td>
<td>flate (ZLIB) compression</td>
</tr>
</tbody>
</table>

Default:

    DOC_COMPRESS_FLATE

Remarks:

When creating a VPE document file using SwapFileName, compression is always activated.

The trial version of VPE will always use compression for exported PDF files.

For the Community Edition, compression is not available.
6.26 OpenFileDialog

[GUI Control Only, not supported by the Community Edition]
Displays a file-open dialog, so the user can select a VPE document file for reading.

```csharp
method void VPE.OpenFileDialog()
```

**Remarks:**

You can preset a file name for the file-open dialog with the property `OpenFileName`. 
6.27 SaveFileDialog

[GUI Control Only, not supported by the Community Edition]
Displays a file-save dialog, so the user can select a file name under which the current VPE document file is written.

method void VPE.SaveFileDialog(
)

Remarks:
You can preset a file name for the file-open dialog with the property SaveFileName.
6.28 OpenFileName

[GUI Control Only, not supported by the Community Edition]

Returns / sets the default path and file name, which will be shown in the Open File Dialog. After the Open File Dialog has been confirmed with OK, this property holds the path and file name selected by the user.

property string VPE.OpenFileName

    read / write; runtime only

Possible Values:
    the file name

Default:
    empty string
6.29 SaveFileName

[GUI Control Only, not supported by the Community Edition]

Returns / sets the default path and file name, which will be shown in the Save File Dialog. After the Save File Dialog has been confirmed with OK, this property holds the path and file name selected by the user.

property string VPE.SaveFileName

read / write; runtime only

Possible Values:
the file name

Default:
empty string
6.30 WriteDoc

Writes the currently open document to an external file.

Depending on the file suffix, the file is written to the following format:

- .vpe – Native VPE document file format
- .pdf – PDF file format
- .htm or .html – HTML file format (requires Professional Edition or higher)
- .xml – XML file format (requires Professional Edition or higher)
- .odt – Open Document Text file format (requires Professional Edition or higher)

For details about creating and using VPE document files, please see the "Programmer's Manual", chapter "Programming Techniques", subchapter "VPE Document Files".

For details about creating PDF files, please see the "Programmer's Manual", chapter "The PDF Export Module".

For details about creating HTML files, please see the "Programmer's Manual", chapter "The HTML Export Module".

```csharp
method boolean VPE.WriteDoc(
    string FileName
)

string FileName
    the path and filename

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Success</td>
</tr>
<tr>
<td>False</td>
<td>Failure</td>
</tr>
</tbody>
</table>

Remarks:

In case of an error, LastError is set.

Depending on the setting of the property DocExportType, you can instruct VPE to write the file to a desired file format (VPE, PDF or HTML), regardless of the supplied file name's suffix.

The property Compression controls, whether a written document (VPE or PDF file format) is compressed.

Protection

VPE Document files can be read, edited and saved using our visual designer dycodoc.
If you want to protect your files from being edited, use the property EditProtection.
6.31 WriteDocPageRange

[Professional Edition and above]

Identical to WriteDoc(), but writes only the given range of pages to a document file.

```csharp
method boolean VPE.WriteDocPageRange(
    string FileName,
    long FromPage,
    long ToPage
)
```

- **string FileName**
  - the path and filename
- **long FromPage**
  - starting page of the page range
- **long ToPage**
  - ending page of the page range

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Success</td>
</tr>
<tr>
<td>False</td>
<td>Failure</td>
</tr>
</tbody>
</table>
6.32 WriteDocStream

[Professional Edition and above]

Identical to WriteDoc(), but writes the document to a stream. Currently the only type of stream offered by VPE is a memory stream.

**method boolean VPE.WriteDocStream(  
   TVPEStream stream
)

*TVPEStream stream*

The stream-object where the document is written to. The stream must have been created before by calling CreateMemoryStream().

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Success</td>
</tr>
<tr>
<td>False</td>
<td>Failure</td>
</tr>
</tbody>
</table>
6.33 WriteDocStreamPageRange

[Professional Edition and above]

Identical to WriteDocStream(), but writes only the given range of pages to a stream.

```csharp
method boolean VPE.WriteDocStreamPageRange(
    TVPEStream stream,
    long FromPage,
    long ToPage
)
```

- **TVPEStream stream**
  The stream-object where the document is written to. The stream must have been created before by calling CreateMemoryStream().

- **long FromPage**
  starting page of the page range

- **long ToPage**
  ending page of the page range

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Success</td>
</tr>
<tr>
<td>False</td>
<td>Failure</td>
</tr>
</tbody>
</table>
6.34 ReadDoc

Reads a VPE or VPE-XML document from the file named <FileName> and appends it to the current document (this even works, if the current document was opened using a Swap File, e.g. it is itself a file!)

The document file may have been created before with OpenDoc() using a Swap File or WriteDoc().

**method boolean VPE.ReadDoc(**
  
  string FileName
)

*string FileName*
the path and filename

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Success</td>
</tr>
<tr>
<td>False</td>
<td>Failure</td>
</tr>
</tbody>
</table>

**Remarks:**
In case of an error, LastError is set.

Tip: If you want to preview or print an already created VPE-Document file, DON'T USE ReadDoc(). Use OpenDoc() with a Swap File instead. This guarantees minimum memory usage with the same performance. (Believe us, our high-performance swapping technology is really FAST.)

This method can **not** import PDF or any other document files, except VPE and VPE-XML document files.
6.35 ReadDocPageRange
[Professional Edition and above]
Identical to ReadDoc(), but reads only the given range of pages from a document file.

method boolean VPE.ReadDocPageRange(
    string FileName,
    long FromPage,    
    long ToPage
)

string FileName
    the path and filename

long FromPage
    starting page of the page range

long ToPage
    ending page of the page range

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Success</td>
</tr>
<tr>
<td>False</td>
<td>Failure</td>
</tr>
</tbody>
</table>

Remarks:
In case of an error, LastError is set.
6.36 ReadDocStream

[Professional Edition and above]

Identical to ReadDoc(), but reads the document from a stream. Currently the only type of stream offered by VPE is a memory stream.

method boolean VPE.ReadDocStream(
   TVPEStream stream
)

TVPEStream stream
   The stream-object where the document is written to. The stream must have been created before by calling CreateMemoryStream() and of course it must hold a valid VPE document.

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Success</td>
</tr>
<tr>
<td>False</td>
<td>Failure</td>
</tr>
</tbody>
</table>
6.37 ReadDocStreamPageRange

[Professional Edition and above]

Identical to ReadDocStream(), but reads only the given range of pages from a stream.

```java
method boolean VPE.ReadDocStreamPageRange(
    TVPEStream stream,
    long FromPage,
    long ToPage
)
```

*TVPEStream stream*

The stream-object where the document is written to. The stream must have been created before by calling CreateMemoryStream() and of course it must hold a valid VPE document.

*long FromPage*

starting page of the page range

*long ToPage*

ending page of the page range

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Success</td>
</tr>
<tr>
<td>False</td>
<td>Failure</td>
</tr>
</tbody>
</table>
### 6.38 DocFileReadOnly

If you use set this property to True, a Document file is opened with read-only permission. A VPE document file can not be created, if this flag is specified. You can only use it to open an existing file for read-only purposes.

If a VPE document file is opened for read / write (the default), no other application can open this file - even if it tries to open the file with read-only permission. Multiple applications can open the file at the same time only, if all applications use `ReadOnly = True`.

If you read an existing document file with `ReadDoc()` into memory, you should set the read-only mode to True. In this case the document file is not opened in exclusive-mode and therefore other applications and users can browse and print it simultaneously.

**property boolean VPE.DocFileReadOnly**

- read / write; design & runtime (read access is only possible, if the document is closed)

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>read-only: file can be shared</td>
</tr>
<tr>
<td>False</td>
<td>Exclusive access</td>
</tr>
</tbody>
</table>

**Default:**

False
6.39 ExternalWindow

[GUI Control Only, not supported by the Community Edition]

Controls, whether the VPE Preview Window is embedded in the parent form it is contained in, or if it is an external window managed by VPE itself.

property boolean VPE.ExternalWindow

read / write; design & runtime, closed document required

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Preview Window is external</td>
</tr>
<tr>
<td>False</td>
<td>Preview Window is embedded</td>
</tr>
</tbody>
</table>

Default:

True

Remarks:

ActiveX: If ExternalWindow is True, it might be necessary to set the Property "Visible" to False, this depends on the container application.

Java: The preview can not be embedded, it is only available as external window.

For the Community Edition the preview can not be embedded, it is only available as external window.
6.40 BorderStyle

[.NET only, GUI Control Only]

If embedded, the style of the border drawn around the control. Use 'None', if fully docked.

**property BorderStyle VPE.BorderStyle**

    read / write; design- and runtime

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed3D</td>
<td>A three-dimensional border</td>
</tr>
<tr>
<td>FixedSingle</td>
<td>A single-line border</td>
</tr>
<tr>
<td>None</td>
<td>No border</td>
</tr>
</tbody>
</table>

**Default:**

None
6.41 Caption

[ActiveX / VCL only, see also the property Text]

If the preview window is external, this will be its caption (i.e. the title of the Preview Window). The caption is also used by VPE, to compose the default JobName of the print job.

**property string VPE.Caption**

read / write; design & runtime, closed document required

**Default:**

"VPE Preview"

**Example:**

```pascal
Doc.Caption = "Monthly Report"
```
6.42 Text

[.NET, Java, PHP, Python, Ruby Only, GUI Control Only]

If the preview window is external, this will be its caption (i.e. the title of the Preview Window). The caption is also used by VPE, to compose the default JobName of the print job.

**property string VPE.Text**

read / write; design & runtime, closed document required

**Default:**
the default caption is generated by the designer, e.g. Visual Studio .Net

**Example:**

```
Doc.Caption = "Monthly Report"
```
6.43 ToolBar

[GUI Control Only, not supported by the Community Edition]

Controls, whether the ToolBar is shown or not.

property boolean VPE.ToolBar

  read / write; design & runtime, closed document required

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the ToolBar</td>
</tr>
<tr>
<td>False</td>
<td>do not show the ToolBar</td>
</tr>
</tbody>
</table>

Default:

  True
6.44 tbOpen

[GUI Control Only, not supported by the Community Edition]

Controls, whether the Open-File Button in the ToolBar is shown or not.

property boolean VPE.tbOpen

   read / write; design & runtime, closed document required

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the Open-File Button</td>
</tr>
<tr>
<td>False</td>
<td>do not show the Open-File Button</td>
</tr>
</tbody>
</table>

Default:

True
6.45 tbSave

[GUI Control Only, not supported by the Community Edition]

Controls, whether the Save-File Button in the ToolBar is shown or not.

**property boolean VPE.tbSave**

read / write; design & runtime, closed document required

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the Save-File Button</td>
</tr>
<tr>
<td>False</td>
<td>do not show the Save-File Button</td>
</tr>
</tbody>
</table>

**Default:**

True
6.46 tbPrint

[GUI Control Only, not supported by the Community Edition]
Controls, whether the Print Button in the ToolBar is shown or not.

property boolean VPE.tbPrint

read / write; design & runtime, closed document required

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the Print Button</td>
</tr>
<tr>
<td>False</td>
<td>do not show the Print Button</td>
</tr>
</tbody>
</table>

Default:
True
6.47 tbPrintSetup

[GUI Control Only, not supported by the Community Edition]

If set to False, no printer setup dialog will appear when the print button (or the corresponding key) in the Preview is pushed.

**property boolean VPE.tbPrintSetup**

read / write; design & runtime (read access is only possible, if the document is closed)

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Enabled</td>
</tr>
<tr>
<td>False</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

**Default:**

enabled (Print Setup Dialog will appear)

For the VpeWebControl, the default is disabled, so no Print Setup Dialog will appear. It is not very meaningful that printer setup dialogs are shown on a running server.
6.48 tbMail

[GUI Control Only, not supported by the Community Edition]

Controls, whether the E-Mail Button in the ToolBar is shown or not.

**property boolean VPE.tbMail**

read / write; design & runtime, closed document required

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the E-Mail Button</td>
</tr>
<tr>
<td>False</td>
<td>do not show the E-Mail Button</td>
</tr>
</tbody>
</table>

**Default:**

True
6.49 EnableMailButton

[GUI Control Only, not supported by the Community Edition]

Setting this property to True / False will enable / disable the e-Mail button in the toolbar.

property boolean VPE.EnableMailButton

write; runtime

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Enabled</td>
</tr>
<tr>
<td>False</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

Default: enabled
6.50 tbScale

[GUI Control Only, not supported by the Community Edition]

Controls, whether the Scale Button Group in the ToolBar is shown or not.

**property boolean VPE.tbScale**

   read / write; design & runtime, closed document required

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the Scale Button Group</td>
</tr>
<tr>
<td>False</td>
<td>do not show the Scale Button Group</td>
</tr>
</tbody>
</table>

**Default:**

   True
6.51 tbGrid

[GUI Control Only, not supported by the Community Edition]
Controls, whether the Grid Button in the ToolBar is shown or not.

property boolean VPE.tbGrid

read / write; design & runtime, closed document required

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the Grid Button</td>
</tr>
<tr>
<td>False</td>
<td>do not show the Grid Button</td>
</tr>
</tbody>
</table>

Default:
False
6.52 GridMode

[GUI Control Only, not supported by the Community Edition]

Specifies, whether the Layout Grid that can be optionally shown in the Preview is drawn in foreground or background. This property does not control, if the grid is drawn. It controls how the grid is drawn.

**property GridMode [integer] VPE/GridMode**

read / write; design- & runtime (read access is only possible, if the document is closed)

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>InForeground</td>
<td>draw the grid in the foreground</td>
</tr>
<tr>
<td>False</td>
<td>InBackground</td>
<td>draw the grid in the background</td>
</tr>
</tbody>
</table>

**Default:**

True (InForeground)
6.53 GridVisible

[GUI Control Only, not supported by the Community Edition]
Specifies, if the Layout Grid is drawn.

property boolean VPE.GridVisible

   read / write; design- & runtime (read access is only possible, if the document is closed)

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>is drawn</td>
</tr>
<tr>
<td>False</td>
<td>is not drawn</td>
</tr>
</tbody>
</table>

Default:
False
6.54 tbNavigate

[GUI Control Only, not supported by the Community Edition]

Controls, whether the Navigation Button Group in the ToolBar is shown or not.

**property boolean VPE.tbNavigate**

read / write; design & runtime, closed document required

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the Navigation Button Group</td>
</tr>
<tr>
<td>False</td>
<td>do not show the Navigation Button Group</td>
</tr>
</tbody>
</table>

**Default:**

True
6.55 tbHelp

[GUI Control Only, not supported by the Community Edition]

Controls, whether the Help-Button in the ToolBar is shown or not.

property boolean VPE.tbHelp

   read / write; design & runtime, closed document required

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the Help Button</td>
</tr>
<tr>
<td>False</td>
<td>do not show the Help Button</td>
</tr>
</tbody>
</table>

Default:

   True
6.56 RouteHelp

[GUI Control Only, not supported by the Community Edition]

If enabled and the Help-Button is visible, pushing the Help-Button or pressing the Help key will cause the event DoHelp() (VCL: OnHelp(), .NET: HelpRequested()) to be sent to the parent window, no matter if the VPE-Preview window is embedded or not. With this option you are able to present context-sensitive help to your user. If this property is False, VPE’s build-in small help dialog will be shown.

**property boolean VPE.RouteHelp**

  read / write; design- & runtime (read access is only possible, if the document is closed)

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>enabled (help event will be sent)</td>
</tr>
<tr>
<td>False</td>
<td>disabled (VPE’s build-in small help dialog will be shown)</td>
</tr>
</tbody>
</table>

**Default:**

  False
6.57 tbAbout

[GUI Control Only, not supported by the Community Edition]

Controls, whether the About Button in the ToolBar is shown or not.

**property boolean VPE.tbAbout**

read / write; design & runtime, closed document required

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the About Button</td>
</tr>
<tr>
<td>False</td>
<td>not show the About Button</td>
</tr>
</tbody>
</table>

**Default:**

True
6.58 tbClose

[GUI Control Only, not supported by the Community Edition]

Controls, whether the Close Button in the ToolBar is shown or not.

If the preview is external (ExternalWindow = True), the close controls of the window (including the system-menu) are also disabled.

**property boolean VPE.tbClose**

read / write; design & runtime, closed document required

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the Close Button</td>
</tr>
<tr>
<td>False</td>
<td>not show the Close Button</td>
</tr>
</tbody>
</table>

**Default:**

True
6.59 EnableCloseButton

[GUI Control Only, not supported by the Community Edition]

Specifies, whether the Close Button in the Preview is enabled (= activated, so the user can push it).

If the preview is external (ExternalWindow = True), the close controls of the window (including the system-menu) are also disabled.

You can use this property to temporarily lock the preview from being closed by the user.

**property boolean VPE.EnableCloseButton**

write; runtime

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>enabled</td>
</tr>
<tr>
<td>False</td>
<td>disabled</td>
</tr>
</tbody>
</table>

**Default:**

enabled (= Close Button is enabled)

**Remarks:**

This property does not control, whether the button is visible in the toolbar or not (see tbClose).
6.60 StatusBar

[GUI Control Only, not supported by the Community Edition]

Controls, whether the StatusBar is shown or not.

property boolean VPE.StatusBar

read / write; design & runtime, closed document required

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the StatusBar</td>
</tr>
<tr>
<td>False</td>
<td>do not show the StatusBar</td>
</tr>
</tbody>
</table>

Default:

True
6.61 PageScroller
[GUI Control Only, not supported by the Community Edition]

Controls, whether the Page Scroller in the Status Bar is shown or not.

property boolean VPE.PageScroller

read / write; design & runtime, closed document required

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the Page Scroller</td>
</tr>
<tr>
<td>False</td>
<td>do not show the Page Scroller</td>
</tr>
</tbody>
</table>

Default:

True
6.62 PageScrollerTracking

[GUI Control Only, not supported by the Community Edition]

When activated (set to true), moving the page scroller in the statusbar will immediately update the preview. Otherwise the preview will be updated if the scrolling has finished.

**property boolean VPE.PageScrollerTracking**

read / write; design- & runtime (read access is only possible, if the document is closed)

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>enabled</td>
</tr>
<tr>
<td>False</td>
<td>disabled</td>
</tr>
</tbody>
</table>

**Default:**

True
6.63 StatusSegment

[GUI Control Only, not supported by the Community Edition]

Controls, whether the Status Segment (where you can show messages and the Progress Bars) in the Status Bar is shown or not.

**property boolean VPE.StatusSegment**

read / write; design & runtime, closed document required

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show the StatusSegment</td>
</tr>
<tr>
<td>False</td>
<td>do not show the StatusSegment</td>
</tr>
</tbody>
</table>

**Default:**

True
6.64 StatusText

[GUI Control Only, not supported by the Community Edition]

Writes the given text into the Statusbar. If text is empty (i.e. ""), the default "Ready" in the current GUI-Language will be displayed.

**property** string VPE.StatusText

write; runtime only

**Possible Values:**
The text that shall be written into the Statusbar.

**Remarks:**
The Progress Bar overlays the text segment in the Statusbar, therefore text will not be visible until the Progress Bar is closed.
6.65 OpenProgressBar

[GUI Control Only, not supported by the Community Edition]

Creates a Progress Bar in the Statusbar with the initial value of zero (0%).

```vpe
method void VPE.OpenProgressBar()
```

Remarks:
The Progress Bar hides the text segment in the Statusbar, therefore text displayed there with `WriteStatusbar()` will not be visible until the Progress Bar is closed.

Example:

```vpe
doc.OpenProgressBar
for n = 0 to 100
    doc.StatusProgress = n
next n
doc.CloseProgressBar
```
### 6.66 StatusProgress

[GUI Control Only, not supported by the Community Edition]

Sets the Progress Bar to the given percentage value.

**property integer VPE.StatusProgress**

write; runtime only

**Possible Values:**

a number between 0 and 100, specifying the percentage value, the progress bar shall display

**Example:**

```plaintext
Doc.OpenProgressBar
for n = 0 to 100
    Doc.StatusProgress = n
next n
Doc.CloseProgressBar
```
6.67 CloseProgressBar
[GUI Control Only, not supported by the Community Edition]
Closes the Progress Bar.

method void VPE.CloseProgressBar(
)

Example:

```
Doc.OpenProgressBar
for n = 0 to 100
    Doc.StatusProgress = n
next n
Doc.CloseProgressBar
```
6.68 BusyProgressBar

[Enhanced Edition and above; GUI Control Only]

When this property is enabled and you are exporting a document, for example to PDF or XML, there will be shown a Progress Bar in the text segment of the Statusbar.

If you are also showing the private Progress Bar (see OpenProgressBar), both Progress Bars will be shown simultaneously. They are drawn in a way that overlapping parts are shown in a mixture of the colors each Progress Bar is using.

property boolean VPE.BusyProgressBar

write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Enabled</td>
</tr>
<tr>
<td>False</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

Default:
True
6.69 Rulers

[GUI Control Only, not supported by the Community Edition]

Controls, whether rulers are shown in the VPE Preview Window or not.

**property boolean VPE.Rulers**

read / write; design & runtime, closed document required

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Rulers are shown</td>
</tr>
<tr>
<td>False</td>
<td>Rulers are not shown</td>
</tr>
</tbody>
</table>

**Default:**

True
### 6.70 RulersMeasure

[GUI Control Only]

Sets the measurement for the rulers. This does not affect the coordinate system of the API. To change the coordinate system of the API, use UnitTransformation.

**property RulersMeasure [long] VPE.RulersMeasure**

read / write; design- & runtime

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CM</td>
<td>0</td>
<td>Centimeter</td>
<td>rulers are shown in metric units</td>
</tr>
<tr>
<td>INCH</td>
<td>1</td>
<td>Inch</td>
<td>rulers are shown in inch units</td>
</tr>
</tbody>
</table>

**Default:**

CM (Centimeter)
6.71 AutoDelete

[GUI Control Only, not supported by the Community Edition]

Specifies, if the document shall be closed (removed from memory) in case the user closes the preview.

property boolean VPE.AutoDelete

    write; runtime

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, close the document also</td>
</tr>
<tr>
<td>False</td>
<td>No</td>
</tr>
</tbody>
</table>

Default:

    True (= the document is deleted if the user closes the preview)
6.72 PreviewWithScrollers

[GUI Control Only, not supported by the Community Edition]

If set to False, the scrollbars in the Preview are forced to be hidden. Use this property with care.

property boolean VPE.PreviewWithScrollers

write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>scrollbars are visible</td>
</tr>
<tr>
<td>False</td>
<td>scrollbars are invisible</td>
</tr>
</tbody>
</table>

Default:

True
6.73 PaperView

[GUI Control Only, not supported by the Community Edition]

If set to True, the document in the preview is drawn like a sheet of paper, reflecting the page-dimensions.

**property boolean VPE.PaperView**

read / write; design- & runtime (read access is only possible, if the document is closed)

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>document is drawn like a sheet of paper</td>
</tr>
<tr>
<td>False</td>
<td>the window background is drawn in white, without borders</td>
</tr>
</tbody>
</table>

**Default:**

True
6.74 Scale

[GUI Control Only, not .NET (see PreviewScale) not supported by the Community Edition]

Returns / sets the scale of the preview (not for printing - see PrintScale).
For .NET Controls the name of this property is reserved.

**property double VPE.Scale**

read / write; runtime

**Possible Values:**
for example: 1.0 = 1:1 0.25 = 1:4 4.0 = 4:1

**Default:**
Variable, the initial scale is computed so that the current page fits with its width into the preview.

**Remarks:**
Not available for .NET, see PreviewScale
For Visual Basic 6 and earlier versions, see VpeScale

**See also:**
   PreviewScale
   VpeScale
6.75 PreviewScale

[GUI Control Only; not supported by the Community Edition]

The same as Scale. Returns / sets the scale of the preview (not for printing - see PrintScale).

**property double VPE.PreviewScale**

  read / write; runtime

**Possible Values:**

  for example: 1.0 = 1:1  0.25 = 1:4  4.0 = 4:1

**Default:**

  Variable, the initial scale is computed so that the current page fits with its width into the preview.

**See also:**

  VpeScale
6.76 VpeScale

[ActiveX only; not supported the Community Edition]

The same as Scale. This method is for use with Visual Basic. Visual Basic has problems with the keywords "Print, Write, Line and Scale". VB doesn't recognize, that these are methods and properties of an ActiveX.

Returns / sets the scale of the preview (not for printing - see PrintScale).

property double VPE.VpeScale

   read / write; runtime

Possible Values:

   for example: 1.0 = 1:1    0.25 = 1:4    4.0 = 4:1

Default:

   Variable, the initial scale is computed so that the current page fits with its width into the preview.
6.77 ScalePercent

[GUI Control Only, not supported by the Community Edition]

Returns / sets the scale for the preview in percent (not for printing - see PrintScale).

property integer VPE.ScalePercent

read / write; runtime

Possible Values:
for example: 100 = 100% (1:1) 25 = 25% (1:4) 400 = 400% (4:1)

Default:
Variable, the initial scale is computed so that the current page fits with its width into the preview.
6.78 MinScale

[GUI Control Only, not supported by the Community Edition]

Specifies the minimum possible preview scale factor that can be set by the user. MinScale may not be less than 0.25 ( = 25 %).

property double VPE.MinScale

write; runtime only

Possible Values:
for example: 1.0 = 1:1 0.25 = 1:4

Default:
0.25
6.79 MinScalePercent

[GUI Control Only, not supported by the Community Edition]

Specifies the minimum preview scale factor that can be set by the user. MinScale may not be less than 25 (25 %)

**property integer VPE.MinScalePercent**

write; runtime only

**Possible Values:**

for example: 100 = 1:1  25 = 1:4

**Default:**

25
6.80 MaxScale

[GUI Control Only, not supported by the Community Edition]

Specifies the maximum preview scale factor that can be set by the user. MaxScale may not be greater than 32.0 (3200 %).

property double VPE.MaxScale

write; runtime only

Possible Values:

  for example: 1.0 = 100%    0.25 = 25%    6.0 = 600%

Default:

  32.0 (3200 %)
6.81 MaxScalePercent

[GUI Control Only, not supported by the Community Edition]

Specifies the maximum preview scale factor that can be set by the user. MaxScale may not be greater than 3200 (3200 %).

**property integer VPE.MaxScalePercent**

  write; runtime only

**Possible Values:**

  for example: 100 = 100%  
  25 = 25%  
  600 = 600%

**Default:**

  3200 (3200 %)
6.82 ScaleMode

[GUI Control Only, not supported by the Community Edition]

Sets or gets the scale mode of the preview.

**property ScaleMode [int] VPE.ScaleMode**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSCALE_MODE_FULL_PAGE</td>
<td>0</td>
<td>FullPage</td>
<td>show full page</td>
</tr>
<tr>
<td>VSCALE_MODE_PAGE_WIDTH</td>
<td>1</td>
<td>PageWidth</td>
<td>fit page width</td>
</tr>
<tr>
<td>VSCALE_MODE_ZOOM_TOOL</td>
<td>2</td>
<td>ZoomTool</td>
<td>zoom tool activated</td>
</tr>
<tr>
<td>VSCALE_MODE_FREE</td>
<td>3</td>
<td>Free</td>
<td>any scale allowed</td>
</tr>
</tbody>
</table>

**Default:**

VSCALE_MODE_PAGE_WIDTH
6.83 ZoomPreview

[GUI Control Only, not supported by the Community Edition]

Zooms a specified rectangle of the current page into the preview.

```java
method void VPE.ZoomPreview(
    long Left,
    long Top,
    long Right,
    long Bottom
)
```

`long Left, Top, Right, Bottom`
the rectangle of the current page in metric or inch coordinates

Remarks:
This method requires a visible preview. If the preview is not visible, the method does nothing.
6.84 ZoomIn

[GUI Control Only, not supported by the Community Edition]

Enlarges the view. Switches to the next larger scale of the internal hard-coded scaling table.

method void VPE.ZoomIn()

Remarks:
This method requires a visible preview. If the preview is not visible, the method does nothing.
6.85 ZoomOut

[GUI Control Only, not supported by the Community Edition]
Reduces the view. Switches to the next smaller scale of the internal hard-coded scaling table.

method void VPE.ZoomOut()

Remarks:
This method requires a visible preview. If the preview is not visible, the method does nothing.
### 6.86 SetPreviewPosition

[GUI Control Only, not supported by the Community Edition]

Scrolls the preview into the given position.

```java
method void VPE.SetPreviewPosition(
    long Left,
    long Top
)
```

*long Left, Top*

the position within the current page in metric or inch coordinates

**Remarks:**

This method requires a visible preview. If the preview is not visible, the method does nothing.
6.87 DefineKey

[GUI Control Only, not supported by the Community Edition]

This method allows you to define your own key for each available keyboard function.

```csharp
method void VPE.DefineKey(
    KeyFunction [long] Function,
    Keys [long] KeyCode,
    Keys [long] AddKeyCode1,
    Keys [long] AddKeyCode2
)
```

`KeyFunction [long] Function`
All available keyboard functions are enumerated in the VKEY_xyz constants (see below).

`Keys [long] KeyCode, AddKeyCode1, AddKeyCode2`
Virtual keycode constants (defined in the Windows SDK) for the keys that need to be pressed at once for the specified function. Add-on keys like Ctrl, Alt and Shift may not be used in the parameter `KeyCode`, use these key codes in the parameters `AddKeyCode1` or `AddKeyCode2`.

Remarks:
A keyboard function can be set to "undefined" = no key is associated with this keyboard function by setting all key code parameters to zero.

Example:

**Active X / VCL:**

```csharp
Doc.DefineKey(VKEY_CLOSE, VK_ESCAPE, 0, 0)
```
Defines the key <ESC> to close the preview when pressed.

```csharp
Doc.DefineKey(VKEY_CLOSE, VK_ESCAPE, VK_SHIFT, 0)
```
Defines the key pair <SHIFT> + <ESC> to close the preview when pressed.

```csharp
Doc.DefineKey(VKEY_PRINT, 0, 0, 0)
```
There is no key associated with the keyboard function VKEY_PRINT.

**.NET:**

```csharp
Doc.DefineKey(KeyFunction.Close, Keys.Escape, 0, 0)
```
Defines the key <ESC> to close the preview when pressed.

```csharp
Doc.DefineKey(KeyFunction.Close, Keys.Escape, Keys.ShiftKey, 0)
```
Defines the key pair <SHIFT> + <ESC> to close the preview when pressed.

```csharp
Doc.DefineKey(KeyFunction.Print, 0, 0, 0)
```
There is no key associated with the keyboard function "Print".
Possible values for the parameter "Function" are:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Enum</th>
<th>Default Key(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VKEY_SCROLL_LEFT</td>
<td>0</td>
<td>ScrollLeft</td>
<td>Arrow Left</td>
</tr>
<tr>
<td>VKEY_SCROLL_PAGE_LEFT</td>
<td>1</td>
<td>ScrollPageLeft</td>
<td>Ctrl-Arrow Left</td>
</tr>
<tr>
<td>VKEY_SCROLL_RIGHT</td>
<td>2</td>
<td>ScrollRight</td>
<td>Arrow Right</td>
</tr>
<tr>
<td>VKEY_SCROLL_PAGE_RIGHT</td>
<td>3</td>
<td>ScrollPageRight</td>
<td>Ctrl-Arrow Right</td>
</tr>
<tr>
<td>VKEY_SCROLL_UP</td>
<td>4</td>
<td>ScrollUp</td>
<td>Arrow Up</td>
</tr>
<tr>
<td>VKEY_SCROLLPAGE_UP</td>
<td>5</td>
<td>ScrollPageUp</td>
<td>Ctrl-Arrow Up</td>
</tr>
<tr>
<td>VKEY_SCROLL_DOWN</td>
<td>6</td>
<td>ScrollDown</td>
<td>Arrow Down</td>
</tr>
<tr>
<td>VKEY_SCROLL_PAGE_DOWN</td>
<td>7</td>
<td>ScrollPageDown</td>
<td>Ctrl-Arrow Down</td>
</tr>
<tr>
<td>VKEY_SCROLL_TOP</td>
<td>8</td>
<td>ScrollTop</td>
<td>Pos1</td>
</tr>
<tr>
<td>VKEY_SCROLLBOTTOM</td>
<td>9</td>
<td>ScrollBottom</td>
<td>End</td>
</tr>
<tr>
<td>VKEY_PRINT</td>
<td>10</td>
<td>Print</td>
<td>F2</td>
</tr>
<tr>
<td>VKEY_MAIL</td>
<td>11</td>
<td>Mail</td>
<td>F3</td>
</tr>
<tr>
<td>VKEY_FULLSCREEN</td>
<td>12</td>
<td>FullPage</td>
<td>Ctrl-Ins</td>
</tr>
<tr>
<td>VKEY_PAGE_WIDTH</td>
<td>13</td>
<td>PageWidth</td>
<td>Ctrl-Del</td>
</tr>
<tr>
<td>VKEY_ZOOM_IN</td>
<td>14</td>
<td>ZoomIn</td>
<td>Ins</td>
</tr>
<tr>
<td>VKEY_ZOOM_OUT</td>
<td>15</td>
<td>ZoomOut</td>
<td>Del</td>
</tr>
<tr>
<td>VKEY_GRID</td>
<td>16</td>
<td>Grid</td>
<td>&quot;G&quot;</td>
</tr>
<tr>
<td>VKEY_PAGE_FIRST</td>
<td>17</td>
<td>PageFirst</td>
<td>Ctrl-Page Up</td>
</tr>
<tr>
<td>VKEY_PAGE_LEFT</td>
<td>18</td>
<td>PageLeft</td>
<td>Page Up</td>
</tr>
<tr>
<td>VKEY_PAGE_RIGHT</td>
<td>19</td>
<td>PageRight</td>
<td>Page Down</td>
</tr>
<tr>
<td>VKEY_PAGE_LAST</td>
<td>20</td>
<td>PageLast</td>
<td>Ctrl-Page Down</td>
</tr>
<tr>
<td>VKEY_HELP</td>
<td>21</td>
<td>Help</td>
<td>F1</td>
</tr>
<tr>
<td>VKEY_INFO</td>
<td>22</td>
<td>Info</td>
<td>&quot;I&quot;</td>
</tr>
<tr>
<td>VKEY_CLOSE</td>
<td>23</td>
<td>Close</td>
<td>&lt;not defined&gt;</td>
</tr>
<tr>
<td>VKEY_GOTO_PAGE</td>
<td>24</td>
<td>GotoPage</td>
<td>ENTER</td>
</tr>
<tr>
<td>VKEY_OPEN</td>
<td>25</td>
<td>Open</td>
<td>Ctrl-O</td>
</tr>
<tr>
<td>VKEY_SAVE</td>
<td>26</td>
<td>Save</td>
<td>Ctrl-S</td>
</tr>
<tr>
<td>VKEY_ESCAPE</td>
<td>27</td>
<td>Escape</td>
<td>Esc</td>
</tr>
</tbody>
</table>

See also:

SendKey
6.88 SendKey

[GUI Control Only, not supported by the Community Edition]

Simulates a keystroke of the user. This allows your application for example to scroll the preview.

**method void VPE.SendKey(**

KeyFunction [long] Vkey

**)

*KeyFunction [long] VKey*

All available keyboard functions are enumerated in the VKEY_xyz constants (see below).

**Example:**

**ActiveX / VCL:**

```
Doc.SendKey(VKEY_SCROLL_DOWN)
```

**.NET:**

```
Doc.SendKey(KeyFunction.ScrollDown)
```

Scrolls the preview down by one cm.

**Possible values for the parameter "VKey" are:**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Enum</th>
</tr>
</thead>
<tbody>
<tr>
<td>VKEY_SCROLL_LEFT</td>
<td>0</td>
<td>ScrollLeft</td>
</tr>
<tr>
<td>VKEY_SCROLL_PAGE_LEFT</td>
<td>1</td>
<td>ScrollPageLeft</td>
</tr>
<tr>
<td>VKEY_SCROLL_RIGHT</td>
<td>2</td>
<td>ScrollRight</td>
</tr>
<tr>
<td>VKEY_SCROLL_PAGE_RIGHT</td>
<td>3</td>
<td>ScrollPageRight</td>
</tr>
<tr>
<td>VKEY_SCROLL_UP</td>
<td>4</td>
<td>ScrollUp</td>
</tr>
<tr>
<td>VKEY_SCROLL_PAGE_UP</td>
<td>5</td>
<td>ScrollPageUp</td>
</tr>
<tr>
<td>VKEY_SCROLL_DOWN</td>
<td>6</td>
<td>ScrollDown</td>
</tr>
<tr>
<td>VKEY_SCROLL_PAGE_DOWN</td>
<td>7</td>
<td>ScrollPageDown</td>
</tr>
<tr>
<td>VKEY_SCROLL_TOP</td>
<td>8</td>
<td>ScrollTop</td>
</tr>
<tr>
<td>VKEY_SCROLL BOTTOM</td>
<td>9</td>
<td>ScrollBottom</td>
</tr>
<tr>
<td>VKEY_PRINT</td>
<td>10</td>
<td>Print</td>
</tr>
<tr>
<td>VKEY_MAIL</td>
<td>11</td>
<td>Mail</td>
</tr>
<tr>
<td>VKEY_FULL_PAGE</td>
<td>12</td>
<td>FullPage</td>
</tr>
<tr>
<td>VKEY_PAGE_WIDTH</td>
<td>13</td>
<td>PageWidth</td>
</tr>
<tr>
<td>VKEY_ZOOM_IN</td>
<td>14</td>
<td>ZoomIn</td>
</tr>
<tr>
<td>VKEY_ZOOM_OUT</td>
<td>15</td>
<td>ZoomOut</td>
</tr>
<tr>
<td>VKEY_GRID</td>
<td>16</td>
<td>Grid</td>
</tr>
<tr>
<td>VKEY_PAGE_FIRST</td>
<td>17</td>
<td>PageFirst</td>
</tr>
<tr>
<td>VKEY_PAGE_LEFT</td>
<td>18</td>
<td>PageLeft</td>
</tr>
<tr>
<td>VKEY_PAGE_RIGHT</td>
<td>19</td>
<td>PageRight</td>
</tr>
<tr>
<td>VKEY</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>VKEY_PAGE_LAST</td>
<td>20</td>
<td>PageLast</td>
</tr>
<tr>
<td>VKEY_HELP</td>
<td>21</td>
<td>Help</td>
</tr>
<tr>
<td>VKEY_INFO</td>
<td>22</td>
<td>Info</td>
</tr>
<tr>
<td>VKEY_CLOSE</td>
<td>23</td>
<td>Close</td>
</tr>
<tr>
<td>VKEY_GOTO_PAGE</td>
<td>24</td>
<td>GotoPage</td>
</tr>
<tr>
<td>VKEY_OPEN</td>
<td>25</td>
<td>Open</td>
</tr>
<tr>
<td>VKEY_SAVE</td>
<td>26</td>
<td>Save</td>
</tr>
<tr>
<td>VKEY_ESCAPE</td>
<td>27</td>
<td>Escape</td>
</tr>
</tbody>
</table>

See also:
- DefineKey
6.89 GUITheme

[GUI Control Only, not supported by the Community Edition]

Specifies the look and feel for the GUI (Graphical User Interface) of the VPE Preview.

**property GUITheme [integer] VPE.GUITheme**

read / write; runtime only

**Possible Values:**

available themes are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUITHEME_OFFICE2000</td>
<td>0</td>
<td>Office2000</td>
<td></td>
</tr>
<tr>
<td>GUITHEME_OFFICE2003</td>
<td>1</td>
<td>Office2003</td>
<td></td>
</tr>
<tr>
<td>GUITHEME_WHIDBEY</td>
<td>2</td>
<td>Whidbey</td>
<td></td>
</tr>
</tbody>
</table>

**Default:**

GUITHEME_WHIDBEY

**Remarks:**

On platforms other than Windows XP, there is no visual difference between the Office2003 and the Whidbey theme. On Windows XP the Office2003 theme uses different color schemes depending on the color theme chosen in the control panel of Windows XP (Blue, Olive or Silver).

On systems with a color resolution below 16-bits it is only possible to select the Office2000 theme.
6.90 GUILanguage

[GUI Control Only]

By default, VPE chooses the language for its GUI (Graphical User Interface) by querying the current chosen language in the control panel of the system. With this property you can override that default behavior and force VPE to use a specific language.

property GUILanguage [integer] VPE.GUILanguage

write; runtime only

Possible Values:

available languages are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGUI_LANGUAGE_USER_DEFINED</td>
<td>-1</td>
<td>UserDefined</td>
<td></td>
</tr>
<tr>
<td>VGUI_LANGUAGE_ENGLISH</td>
<td>0</td>
<td>English</td>
<td></td>
</tr>
<tr>
<td>VGUI_LANGUAGE_GERMAN</td>
<td>1</td>
<td>German</td>
<td></td>
</tr>
<tr>
<td>VGUI_LANGUAGE_FRENCH</td>
<td>2</td>
<td>French</td>
<td></td>
</tr>
<tr>
<td>VGUI_LANGUAGE_DUTCH</td>
<td>3</td>
<td>Dutch</td>
<td></td>
</tr>
<tr>
<td>VGUI_LANGUAGE_SPANISH</td>
<td>4</td>
<td>Spanish</td>
<td></td>
</tr>
<tr>
<td>VGUI_LANGUAGE_DANISH</td>
<td>5</td>
<td>Danish</td>
<td></td>
</tr>
<tr>
<td>VGUI_LANGUAGE_SWEDISH</td>
<td>6</td>
<td>Swedish</td>
<td></td>
</tr>
<tr>
<td>VGUI_LANGUAGE_FINNISH</td>
<td>7</td>
<td>Finnish</td>
<td></td>
</tr>
<tr>
<td>VGUI_LANGUAGE_ITALIAN</td>
<td>8</td>
<td>Italian</td>
<td></td>
</tr>
<tr>
<td>VGUI_LANGUAGE_NORWEGIAN</td>
<td>9</td>
<td>Norwegian</td>
<td></td>
</tr>
<tr>
<td>VGUI_LANGUAGE_PORTUGUESE</td>
<td>10</td>
<td>Portuguese</td>
<td></td>
</tr>
</tbody>
</table>

Default:

depends on the chosen language in the control panel of the system

Remarks:

If you set the GUILanguage to UserDefined, you can specify freely any text in any language for each element of the GUI by calling SetResourceString.

Example:

**ActiveX / VCL:**

```
Doc.GUILanguage = VGUI_LANGUAGE_SPANISH
```

**.NET:**

```
Doc.GUILanguage = GUILanguage.Spanish
```

forces VPE to use spanish as language for the GUI
**6.91 SetResourceString**

*GUI Control Only, not supported by the Community Edition*

If you set the `GUILanguage` to `UserDefined`, you can specify freely any text in any language for each element of the GUI.

```method void VPE.SetResourceString(
    ResourceId [int] Id,
    string Text
)
```

*ResourceId [int] Id*

possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRSCID_COPY_OF</td>
<td>0</td>
<td>CopyOf</td>
<td>&quot;Copy %u of %u&quot;</td>
</tr>
<tr>
<td>VRSCID_PAGE_OF</td>
<td>1</td>
<td>PageOf</td>
<td>&quot;Page %u of %u&quot;</td>
</tr>
<tr>
<td>VRSCID_BAND</td>
<td>2</td>
<td>Band</td>
<td>&quot;Band %u&quot; (i.e. &quot;stripe&quot; = &quot;band&quot;)</td>
</tr>
<tr>
<td>VRSCID_ERROR</td>
<td>3</td>
<td>Error</td>
<td>&quot;Error&quot;</td>
</tr>
<tr>
<td>VRSCID_WARNING</td>
<td>4</td>
<td>Warning</td>
<td>&quot;Warning&quot;</td>
</tr>
<tr>
<td>VRSCID_PRINTING</td>
<td>5</td>
<td>Printing</td>
<td>&quot;Printing&quot;</td>
</tr>
<tr>
<td>VRSCID_CANCEL</td>
<td>6</td>
<td>Cancel</td>
<td>&quot;Cancel&quot;</td>
</tr>
<tr>
<td>VRSCID_OK</td>
<td>7</td>
<td>Ok</td>
<td>&quot;Ok&quot;</td>
</tr>
</tbody>
</table>

*Tooltips:*

<p>| VRSCID_PRINT_DOC       | 8     | PrintDoc | &quot;Print the Document&quot;  |
| VRSCID_FULL_PAGE       | 9     | FullPage | &quot;Show Full Page&quot;       |
| VRSCID_PAGE_WIDTH      | 10    | PageWidth| &quot;Fit Page Width&quot;       |
| VRSCID_ZOOM_IN         | 11    | ZoomIn   | &quot;Zoom in&quot;              |
| VRSCID_ZOOM_OUT        | 12    | ZoomOut  | &quot;Zoom out&quot;             |
| VRSCID_FIRST_PAGE      | 13    | FirstPage| &quot;Go to First Page&quot;     |
| VRSCID_PREV_PAGE       | 14    | PrevPage | &quot;Go to Previous Page&quot;  |
| VRSCID_NEXT_PAGE       | 15    | NextPage | &quot;Go to Next Page&quot;      |
| VRSCID_LAST_PAGE       | 16    | LastPage | &quot;Go to Last Page&quot;      |
| VRSCID_HELP            | 17    | Help     | &quot;Help on Control&quot;      |
| VRSCID_INFORMATION     | 18    | Information| &quot;Program Information&quot; |
| VRSCID_CLOSE_PREVIEW   | 19    | ClosePreview| &quot;Close Preview&quot;        |
| VRSCID_GRID            | 20    | Grid     | &quot;Show / Hide Grid&quot;     |
| VRSCID_ENTER_PAGENO    | 21    | EnterPageno| &quot;enter page# with mouse-click&quot; |
| VRSCID_ZOOM_FACTOR     | 22    | ZoomFactor| &quot;Zoom Factor&quot;          |</p>
<table>
<thead>
<tr>
<th>Resource Code</th>
<th>ID</th>
<th>Type</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRSCID_STATUS</td>
<td>23</td>
<td>Status</td>
<td>&quot;Status&quot;</td>
</tr>
<tr>
<td>VRSCID_READY</td>
<td>24</td>
<td>Ready</td>
<td>&quot;Ready&quot;</td>
</tr>
<tr>
<td>VRSCID_EMAIL</td>
<td>25</td>
<td>Email</td>
<td>&quot;eMail&quot;</td>
</tr>
<tr>
<td>VRSCID_OPEN</td>
<td>26</td>
<td>Open</td>
<td>&quot;Open&quot;</td>
</tr>
<tr>
<td>VRSCID_SAVE</td>
<td>27</td>
<td>Save</td>
<td>&quot;Save&quot;</td>
</tr>
<tr>
<td>VRSCID_ERROR_OPEN</td>
<td>28</td>
<td>ErrorOpen</td>
<td>&quot;Could not open file!&quot;</td>
</tr>
<tr>
<td>VRSCID_ERROR_WRITE</td>
<td>29</td>
<td>ErrorWrite</td>
<td>&quot;Could not write file!&quot;</td>
</tr>
<tr>
<td>VRSCID_USAGE</td>
<td>30</td>
<td>Usage</td>
<td>&quot;Usage&quot; (caption of the help-dialog)</td>
</tr>
<tr>
<td>VRSCID_MOUSE</td>
<td>31</td>
<td>Mouse</td>
<td>&quot;Mouse&quot;</td>
</tr>
<tr>
<td>VRSCID_USAGE_MOUSE</td>
<td>32</td>
<td>UsageMouse</td>
<td>&lt;all mouse usage text, lines separated by carriage return or linefeed&gt;</td>
</tr>
<tr>
<td>VRSCID_KEYBOARD</td>
<td>33</td>
<td>Keyboard</td>
<td>&quot;Keyboard&quot;</td>
</tr>
<tr>
<td>VRSCID_USAGE_KEYBOARD</td>
<td>34</td>
<td>UsageKeyboard</td>
<td>&lt;all keyboard usage text, lines separated by carriage return or linefeed&gt;</td>
</tr>
</tbody>
</table>

**Other:**

<table>
<thead>
<tr>
<th>Resource Code</th>
<th>ID</th>
<th>Type</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRSCID_USAGE</td>
<td>30</td>
<td>Usage</td>
<td>&quot;Usage&quot; (caption of the help-dialog)</td>
</tr>
<tr>
<td>VRSCID_MOUSE</td>
<td>31</td>
<td>Mouse</td>
<td>&quot;Mouse&quot;</td>
</tr>
<tr>
<td>VRSCID_USAGE_MOUSE</td>
<td>32</td>
<td>UsageMouse</td>
<td>&lt;all mouse usage text, lines separated by carriage return or linefeed&gt;</td>
</tr>
<tr>
<td>VRSCID_KEYBOARD</td>
<td>33</td>
<td>Keyboard</td>
<td>&quot;Keyboard&quot;</td>
</tr>
<tr>
<td>VRSCID_USAGE_KEYBOARD</td>
<td>34</td>
<td>UsageKeyboard</td>
<td>&lt;all keyboard usage text, lines separated by carriage return or linefeed&gt;</td>
</tr>
</tbody>
</table>

**string Text**
The text that shall be used for the specified resource.
6.92 hWndPreview

[GUI Control Only, .NET, ActiceX, VCL only]

Returns the window handle of the Preview. This is a standard window handle you can use for manipulating the Preview. It is also needed when working with an embedded Preview Window, to control the position and the size of the Preview and for routing (sending) keyboard-messages (WM_CHAR) to it.

**property IntPtr [long] VPE.hWndPreview**

read; runtime only

**Returns:**

the window handle of the preview
6.93 PreviewWindow

[.NET only, GUI Control Only]

Returns the IWin32Window of the Preview. It can be used directly as parameter for many .NET methods, like for example MessageBox.Show().

property PreviewHandle VPE.PreviewWindow

    read; runtime only

Returns:

the IWin32Window of the preview

PreviewHandle is defined as:

```csharp
public class PreviewHandle : IWin32Window
```
6.94 ControlVersion
Returns the current version number of the ActiveX / VCL / .NET component.

property double VPE-ControlVersion
  read; runtime only
Returns:
The current version number.
6.95 VpeVersion

Returns the current version number of the loaded VPE-DLL.

property double VPE.VpeVersion

read; runtime only

Returns:
The current version number.
6.96 Edition

Returns the Edition of the loaded Engine.


read; runtime only

**Returns:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VEDITION_COMMUNITY</td>
<td>500</td>
<td>Community</td>
<td></td>
</tr>
<tr>
<td>VEDITION_STANDARD</td>
<td>1000</td>
<td>Standard</td>
<td></td>
</tr>
<tr>
<td>VEDITION_ENHANCED</td>
<td>2000</td>
<td>Enhanced</td>
<td></td>
</tr>
<tr>
<td>VEDITION_PROFESSIONAL</td>
<td>3000</td>
<td>Professional</td>
<td></td>
</tr>
<tr>
<td>VEDITION_ENTERPRISE</td>
<td>4000</td>
<td>Enterprise</td>
<td></td>
</tr>
<tr>
<td>VEDITION_INTERACTIVE</td>
<td>5000</td>
<td>Interactive</td>
<td></td>
</tr>
</tbody>
</table>
7 Printing Functions

7.1 Printing Functions
These functions deal with setting-up the printer, controlling print-options and printing itself.

7.2 SetupPrinter
[Windows platform only; not supported by PHP and the Community Edition]
This is a very central function for controlling the printer. You can create a setup for the printer separate from printing. The settings for the printer can be stored/retrieved in/from a file (optional), so the settings can be used permanently.

The method is very flexible, as your end-user may make individual printer-setups for each kind of document. So the user can not only specify an individual printer for each different type of document, but also the printer's properties, like the number of copies, collation, output paper bin and everything else. Just let the user make these definitions in an extra part of your application and store the different settings into different files. This will give you the best control possible, and you can easily implement a "Printer Setup" function in your applications menu using this method.

The setup and storage into a file is done with a single call to SetupPrinter(), where you provide the file name as parameter and set the parameter DialogControl = PRINTDLG_ALWAYS.
For example with the call: SetupPrinter("document_type_1.prs", PRINTDLG_ALWAYS)
VPE will try to read the file "document_type_1", and load it, if it exists. Next, a dialog with the settings stored in the file (or default settings, if no file is found) will appear and your user can make all settings available in the dialog(s). If the user then pushes the OK button, the settings are stored in the file "document_type_1.prs".

Later, when printing the document of "type 1", you call OpenDoc() and then SetupPrinter("document_type_1.prs", PRINTDLG_ONFAIL). The flag PRINTDLG_ONFAIL will let the setup-dialog only come up, if the file "document_type_1.prs" is not found. For example if your user hasn't done the setup yet (the specified file is not found), it will be done then and only once! Otherwise, if the file is found, no dialog is shown and the printer setup - using all previously stored settings from the file - is performed "silently".

Important: VPE saves all settings to the setup file: name of the printer, driver, port and in addition all other driver-specific private data. The private printer-driver data includes every special option a printer-driver provides, even those which can not be accessed through the standard Windows API. This means that your users can set, store and use just any option of the printer!
method integer VPE.SetupPrinter(
    string FileName,
    PrintDialogControl [integer] DialogControl
)

string FileName
    (path and) file name of the file where to restore or store the setup or ""

PrintDialogControl [integer] DialogControl
possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINTDLG_NEVER</td>
<td>0</td>
<td>Never</td>
<td>never show setup-dialog (if file_name is &quot;&quot;, the last setting or the setting of the default-printer will be taken)</td>
</tr>
<tr>
<td>PRINTDLG_ONFAIL</td>
<td>1</td>
<td>OnFail</td>
<td>show setup-dialog only, if file-read fails</td>
</tr>
<tr>
<td>PRINTDLG_ALWAYS</td>
<td>2</td>
<td>Always</td>
<td>show setup-dialog always</td>
</tr>
<tr>
<td>PRINTDLG_FULL</td>
<td>4</td>
<td>Full</td>
<td>show FULL Dialog (with page range, collation, etc.). This is a flag, i.e. add it to one of the other PRINTDLG_xyz flags (example: PRINTDLG_ALWAYS + PRINTDLG_FULL). The flag is especially useful if you hide the Toolbar or the Print-Button and you want to provide to your end-user the possibility to make a full printer setup including a page-range, etc.</td>
</tr>
</tbody>
</table>

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Ok</td>
</tr>
<tr>
<td>1</td>
<td>User pushed cancel button in dialog</td>
</tr>
<tr>
<td>2</td>
<td>I/O problems during read of setup file (only if dialog_control = PRINTDLG_NEVER = 0)</td>
</tr>
<tr>
<td>3</td>
<td>I/O problems during write of setup file</td>
</tr>
<tr>
<td>4</td>
<td>problem in the printing system, for example the printer used in the setup file is no longer installed in the system</td>
</tr>
</tbody>
</table>

Remarks:
For standardization, Printer Setup Files created with VPE should have the suffix ".PRS".

Beside using SetupPrinter(), you can read Printer Setup Files into VPE by using ReadPrinterSetup() and you can write them using WritePrinterSetup().

If there is a problem in the printing system, for example the printer used in the setup file is no longer installed in the system, SetupPrinter() will try to select the default printer. If this is not possible, SetupPrinter() will return the value "4".

VPE is not using the settings of a printer for the document (otherwise it wouldn't be printer-independent). In order to reflect the settings of the printer, let the user make a printer setup - or read an existing setup from a PRS file - and assign the Device Control Properties like DevPaperFormat and DevOrientation, etc. to the current VPE Document (see PageFormat and PageOrientation). For an example, see below.
The page range is not written to a PRS File:
It does not make sense to store the page range within a generic setup file, which is intended to be used for different documents.

The settings in the PRS file for the page orientation and page format are ignored:
The **PageOrientation** (Landscape / Portrait) chosen in the Printer Setup Dialog is ignored by VPE: this is by design, because VPE gives you - the developer - the control over the printout. That means: because you can set in a VPE Document the page orientation for each page separately by code, the printer setting for the orientation will be overridden by the settings in the VPE document during printing.

The same rule applies to the **PageFormat** (or individual page dimensions): since you can specify different page formats for each page in a VPE Document separately, VPE ignores the settings in a PRS file and uses the settings of the document. This has the advantage, that VPE instructs the printer at each newly printed page, to use the page dimensions of the currently printed page. Many printers can react on this and choose automatically the right paper from a different paper input bin. If you want to stop VPE from this automatism, or if you need to retrieve the page format from the PRS file, you can specify the flag PRINT_NO_AUTO_PAGE_DIMS for the property **PrintOptions**.

Example:

**ActiveX / VCL:**

```plaintext
VPE.OpenDoc
// use PRINT_NO_AUTO_PAGE_DIMS, in order to be able to retrieve the values
// for the page dimensions from the PRS file:
VPE.PrintOptions = PRINT_ALL + PRINT_NO_AUTO_PAGE_DIMS
VPE.SetupPrinter "personal settings.prs", PRINTDLG_ONFAIL
VPE.PageWidth = VPE.DevPaperWidth
VPE.PageHeight = VPE.DevPaperHeight
VPE.PageOrientation = VPE.DevOrientation

// switch PrintOptions back and let VPE control the printer's page
// dimensions, i.e. clear the flag PRINT_NO_AUTO_PAGE_DIMS
VPE.PrintOptions = PRINT_ALL
```

**.NET:**

```plaintext
VPE.OpenDoc
// use NoAutoPageDims, in order to be able to retrieve the values
// for the page dimensions from the PRS file:
VB:  VPE.PrintOptions = PrintOptions.PrintAll + PrintOptions.NoAutoPageDims
C#:  VPE.PrintOptions = PrintOptions.PrintAll | PrintOptions.NoAutoPageDims
VPE.SetupPrinter "personal settings.prs", PrintDialogControl.OnFail
VPE.PageWidth = VPE.DevPaperWidth
VPE.PageHeight = VPE.DevPaperHeight
VPE.PageOrientation = VPE.DevOrientation

// switch PrintOptions back and let VPE control the printer's page
// dimensions, i.e. clear the flag PRINT_NO_AUTO_PAGE_DIMS
VPE.PrintOptions = PrintOptions.PrintAll
```

You will notice that we are using **PageWidth** and **PageHeight** in the example above. The reason is, that this circumvents a bug in many printer drivers, which return wrong values for DevPaperFormat in case a user defined format has been assigned to the printer. Because of this, the construction "VPE.PageFormat = VPE.DevPaperFormat" can not be used reliably. Instead, use the code from the example above.

**Printing to a tractor printer using endless paper / labels:**
The Windows operating system always generates a Form Feed after a page has been printed. In order to print on endless paper or labels with a tractor printer, set the PageFormat or PageWidth and PageHeight of the VPE Document correspondingly to the dimensions of the paper / label. Since VPE will instruct the printer to use the document's page format, this will force the printer to position exactly on the next page/label after a single page/label has been printed.
7.3 PrintOptions

[Windows platform only; not supported by PHP and the Community Edition]

Allows to set several print options. This preset is also active if the user pushes the print-button in the toolbar, or if the method SetupPrinter() is called.

**property PrintOptions [long] VPE.PrintOptions**

write; runtime only

**Possible Values:**

A combination of the following values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT_ALL</td>
<td>0</td>
<td>PrintAll</td>
<td>print all pages</td>
</tr>
<tr>
<td>PRINT_EVEN</td>
<td>1</td>
<td>PrintEven</td>
<td>print only even pages</td>
</tr>
<tr>
<td>PRINT_ODD</td>
<td>2</td>
<td>PrintOdd</td>
<td>print only odd pages</td>
</tr>
<tr>
<td>PRINT_NOABORTDLG</td>
<td>4</td>
<td>NoAbortDialog</td>
<td>show no abort / progress dialog during printing</td>
</tr>
<tr>
<td>PRINT_NO_AUTO_PAGE_DIMS</td>
<td>16</td>
<td>NoAutoPageDims</td>
<td>VPE shall not instruct the printer to use the page dimensions of the currently printed page</td>
</tr>
</tbody>
</table>

Default: PRINT_ALL

For the VpeWebControl it is: PrintAll + NoAbortDialog, because it is not meaningful to show a progress / abort dialog on a server.

**Example:**

The several different options can be set by adding the values of each flag. When you assign a new value to PrintOptions, any previous settings will be overwritten:

**ActiveX / VCL:**

```csharp
Doc.PrintOptions = PRINT_EVEN
```

print only even numbered pages

```csharp
Doc.PrintOptions = PRINT_ODD + PRINT_NOABORTDLG
```

print only odd numbered pages and do not show the print-progress dialog

```csharp
Doc.PrintOptions = PRINT_ALL + PRINT_NOABORTDLG + PRINT_NO_AUTO_PAGE_DIMS
```

print all pages, do not show the print-progress dialog, do not instruct the printer to use the page dimensions of the currently printed page

```csharp
Doc.PrintOptions = PRINT_ALL
```

reset to default

**.NET:**

```csharp
Doc.PrintOptions = PrintOptions.PrintEven
```

print only even numbered pages
Doc.PrintOptions = PrintOptions.PrintOdd + PrintOptions.NoAbortDialog

print only odd numbered pages and do not show the print-progress dialog

Doc.PrintOptions = PrintOptions.PrintAll + PrintOptions.NoAbortDialog +
                   PrintOptions.NoAutoPageDims

print all pages, do not show the print-progress dialog, do not instruct the printer to use the
page dimensions of the currently printed page

Doc.PrintOptions = PrintOptions.PrintAll

reset to default

C#:
Doc.PrintOptions = PrintOptions.PrintEven

print only even numbered pages

Doc.PrintOptions = PrintOptions.PrintOdd | PrintOptions.NoAbortDialog

print only odd numbered pages and do not show the print-progress dialog

Doc.PrintOptions = PrintOptions.PrintAll | PrintOptions.NoAbortDialog |
                   PrintOptions.NoAutoPageDims

print all pages, do not show the print-progress dialog, do not instruct the printer to use the
page dimensions of the currently printed page

Doc.PrintOptions = PrintOptions.PrintAll

reset to default

Remarks:

PRINT_NO_AUTO_PAGE_DIMS:
By default, VPE instructs the printer during printing, to switch the paper dimensions
automatically according to the page dimensions of the VPE Document's currently printed page.
Many printers can react on this and choose automatically the right paper from a different paper
input bin. If you want to stop VPE from this automatism, you can specify the flag
PRINT_NO_AUTO_PAGE_DIMS for this property.

Use PRINT_NO_AUTO_PAGE_DIMS also, if you require that the SetupPrinter() method
retrieves the settings for the page dimensions from a PRS file.

It has been reported, that some very few printer drivers have a bug and do not print in duplex
mode, unless you use PRINT_NO_AUTO_PAGE_DIMS.

In order to let the printer accept the automatic selection of user defined page formats (i.e. you
are not specifying PRINT_NO_AUTO_PAGE_DIMS), it might be necessary to define a form of
the desired page format.

Example: with "Start Menu | Settings | Printers" the window with the installed printers will
appear. Right-click on a blank area of the window and choose "Server Properties" from the
pop-up menu. In the upcoming dialog, define a custom form of the desired dimensions. For
example name it "Test" and set the width to 760 and height to 1280.
In your source code, select the printer and the desired page format like this:
VPE.Device = "Epson LQ-550"
VPE.PageWidth = 760
VPE.PageHeight = 1280

That's it.
7.4 PrintPosMode

[Windows platform only; not supported by PHP and the Community Edition]

Specifies, how objects are positioned on the printer.

property PrintPosMode [integer] VPE.PrintPosMode

write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINTPOS_ABSOLUTE</td>
<td>0</td>
<td>Absolute</td>
<td>absolute to the paper-edges (default)</td>
</tr>
<tr>
<td>PRINTPOS_RELATIVE</td>
<td>1</td>
<td>Relative</td>
<td>relative to the first printable position of the printer</td>
</tr>
</tbody>
</table>

Default:
PRINTPOS_ABSOLUTE

See also:
"Positioning On the Printer" in the Programmer’s Manual and SetPrintOffset
7.5 SetPrintOffset

[Windows platform only; not supported by PHP and the Community Edition]

Allows to set a printing offset by code. This offset will be added to all objects in the document while printing. It is especially useful for calibrating printers that return wrong information about their unprintable area.

method void VPE.SetPrintOffset(
    VpeCoord OffsetX,
    VpeCoord OffsetY
)

VpeCoord OffsetX, OffsetY
    the offset in metric or inch units, values can be positive or negative

Default:
    OffsetX  = 0  (no offset)
    OffsetY  = 0  (no offset)

See also:
    "Positioning On the Printer" in the Programmer's Manual and
    PrintPosMode
7.6 PrintOffsetX

[Windows platform only; not supported by PHP and the Community Edition]

Allows to set a printing offset by code. This offset will be added to all objects in the document while printing. It is especially useful for calibrating printers that return wrong information about their unprintable area.

property VpeCoord VPE.PrintOffsetX

    read / write; runtime only

Possible Values:

the x-offset in metric or inch units, values can be positive or negative

Default:

0 (no offset)

See also:

"Positioning On the Printer" in the Programmer's Manual and PrintPosMode
7.7 PrintOffsetY

[Windows platform only; not supported by PHP and the Community Edition]

Allows to set a printing offset by code. This offset will be added to all objects in the document while printing. It is especially useful for calibrating printers that return wrong information about their unprintable area.

property VpeCoord VPE.PrintOffsetY

   read / write; runtime only

Possible Values:
   the y-offset in metric or inch units, values can be positive or negative

Default:
   0 (no offset)

See also:
   "Positioning On the Printer" in the Programmer's Manual and PrintPosMode
7.8 PrintScale

[Windows platform only, Professional Edition and higher only, not supported by PHP]

This property sets the scale for printing. It does not affect the scale in the preview.

Since the X- and Y- dimensions are scaled simultaneously, this property changes the width and the height of the printout at the same time. The change of the PrintScale implies a change of the positions of all object's contained in the document.

This might lead to problems regarding the positioning of the topmost / leftmost objects:
if for example a text is placed at the position (2, 2) and the PrintScale is set to 0.75 then the position of the text is changed to 0.75 * (2, 2) = (1.5, 1.5). If you have placed objects very near to the unprintable area, or if you have set the PrintScale to a very small value, objects might be moved into the unprintable area and clipped.

This can be corrected by setting the PrintOffsetX- and PrintOffsetY - properties. By using these properties you can align the printout anywhere on the paper, even on the center.

The PrintScale property is not stored in VPE Document files. Therefore VPEView - the Document Viewer - will print all documents unscaled.

property double VPE.PrintScale

read / write; runtime only

Default:
1.0 which means the printout is unscaled (the scaling factor is 1:1)

Remarks:
The setting for the PrintScale is not stored in VPE Document files.

Example:

VPE.PrintScale = 0.5

Sets the print scale to 0.5 : 1, i.e. the printout is half the size than the original size.

VPE.PrintScale = 4.0

Sets the print scale to 4 : 1, i.e. the printout is four times bigger than the original size.

See also:
7.9 PrintDoc

[Windows platform only; not supported by PHP and the Community Edition]

Prints the document. You may not close your application or the document until VPE has finished all print-jobs. Your application code will hold at the PrintDoc() call as long as all pages are printed. But your application will still be able to receive window messages.

During printing, VPE will disable the parent window of the preview supplied as parameter in OpenDoc(). If the parent window is not the main window of your application, or if your application is in any other way able to receive and to respond to messages, it is your responsibility to disable your application, or to take care that your print-functions (for example a button or a menu entry that invokes printing) are stopped from being reentered.

**method void VPE.PrintDoc(**

  **boolean WithSetup**

  **)***

**boolean WithSetup**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show printer setup-dialog</td>
</tr>
<tr>
<td>False</td>
<td>do not show printer setup-dialog</td>
</tr>
</tbody>
</table>

**Remarks:**

After calling PrintDoc(), code execution is suspended until the document has been completely printed. So the following command sequence is absolutely valid:

```c
Call Report.PrintDoc(True or False)
Call Report.CloseDoc
```

The document will be closed AFTER it has been printed (or, if the print has been aborted by the user, which can be realized with the "Print" event). But due to the event-driven mechanism of Windows, your form is still "alive" and other actions can take place (like button clicks), and the related code will be executed.
7.10 IsPrinting

[Windows platform only; not supported by PHP]

If your program, or the user, started printing (by clicking the toolbar button), this function will return True. While printing, the document may not be modified, nor closed.

property boolean VPE.IsPrinting

   read; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>the Engine is currently printing</td>
</tr>
<tr>
<td>False</td>
<td>the Engine is currently not printing</td>
</tr>
</tbody>
</table>
8 Device Control Properties

8.1 Device Control Properties

[Windows platform only; not supported by PHP and the Community Edition]

VPE offers a large set of properties and methods to give you all control over the printing device. By default, VPE selects now the default (standard) output device after a call to OpenDoc().

Remarks:

- Changing the properties (like DevBin, DevOrientation, DevPaperFormat, etc.) during the print-job doesn’t work with some (buggy) printer drivers, for example some fax drivers and the HP4 M Plus driver (the HP4 PS driver should work with the HP4 M Plus printer!).

- Some properties and methods don’t work with some printer drivers. For example "DevTTOption" doesn’t work with our HP4 and HP5 printer drivers on WfW 3.11 and NT 3.51, but it works with both drivers on Win95. This is a driver problem.

- Printer drivers are manufactured by vendors independent of IDEAL Software; we make no warranty, implied or otherwise, regarding these product’s performance or reliability.

- Win32s is not officially supported by VPE. The Sophisticated Printer Control does not work with Win32s.
8.2 DevEnum

[Windows platform only; not supported by PHP and the Community Edition]

Initialize enumeration of all available output devices. After this method has been called, you can retrieve each single device name with the method GetDevEntry().

```java
method long VPE.DevEnum()
```

**Returns:**
the number of printing devices installed in the system

**Remarks:**
Do not call DevEnum() in a loop like "for x = 0 to Doc.DevEnum() - 1", because each time VPE will check for all available devices, which is time-consuming. Rather assign the value of DevEnum() to a variable.

**Example:**
see GetDevEntry()
8.3 GetDevEntry

[Windows platform only; not supported by PHP and the Community Edition]

After calling DevEnum(), you are able to retrieve the names of all printing devices installed on the system with this method. With each call you retrieve one entry.

```
method string VPE.GetDevEntry(
    long Index
)
```

`long Index` must be in the range between 0 and the value returned DevEnum() - 1

**Returns:**

The name of the device specified by "index". If index is out of range, an empty string ("") will be returned.

**Remarks:**

This method works only correctly, if you called DevEnum() before.

**Example:**

```csharp
long i, count
string s
count = DevEnum() - 1 // initialize enumeration
for i = 0 to count
    s = DevEntry(i)
    MessageBox(s) // show each available device in a separate message box
next i
```

**Note:**

Do not call DevEnum() repeatedly, because each time VPE will query all printers from the system.

**Wrong Example:**

```csharp
long i
string s
for i = 0 to DevEnum() - 1
    s = DevEntry(i)
    MessageBox(s) // show each available device in a separate message box
next i
```
8.4 Device

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets the current output device. After a new output device is successfully selected, all device settings (like DevPaperFormat, etc.) are lost! You are also able to select the default output device by setting device to an empty string ("").

**property string VPE.Device**

read / write; runtime only

**Possible Values:**
the name of the output device

**Remarks:**
In case of an error, LastError is set to VERR_COMMON. An error may occur in case the specified device is not present or not available for some reason.

You can enumerate the available devices with DevEnum().

**Example:**

```pascal
Doc.Device = "Office Printer"
```

Selects the device installed as "Office Printer" in the system.

```pascal
Doc.Device = ""
```

Selects the default device of the system.
8.5 DevOrientation

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets the orientation of the currently selected output device to portrait or landscape. Setting this property does NOT affect the setting of the preview (see PageOrientation).

**property PageOrientation [integer] VPE.DevOrientation**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VORIENT_PORTRAIT</td>
<td>1</td>
<td>Portrait</td>
<td></td>
</tr>
<tr>
<td>VORIENT_LANDSCAPE</td>
<td>2</td>
<td>Landscape</td>
<td></td>
</tr>
</tbody>
</table>

**Remarks:**

The DevOrientation property is available for total control. It is only useful if called while processing the BeforePrintNewPage() event (VCL: OnPrintNewPage(), .NET: BeforePrintNewPage()). Otherwise the DevOrientation property is always overridden by the PageOrientation property. You should always use the PageOrientation property to change the orientation for the preview and the printer.

In case of an error, LastError is set to VERR_COMMON. An error may occur if the device does not support changing the orientation.

**Example:**

**ActiveX / VCL**

```
Doc.DevOrientation = VORIENT_LANDSCAPE
```

**.NET**

```
Doc.DevOrientation = PageOrientation.Landscape
```
8.6 DevPaperFormat

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets the paper format of the currently selected output device to one of the predefined VPAPER_xyz formats. You may only use one of the VPAPER_xyz constants as a valid value. Setting this property does NOT affect the setting of the preview (see PageFormat).

Setting this property instructs a printer with multiple paper bins to select the input bin which contains paper of the appropriate paper format. This property is also useful for tractor printers with endless paper feeders.

property integer VPE.DevPaperFormat

    read / write; runtime only

Possible Values:
    one of the predefined VPAPER_xyz constants (see below)

Remarks:
    This property is available for total control. It is only useful if called while processing the BeforePrintNewPage() event (VCL: OnPrintNewPage(), .NET: BeforePrintNewPage()). Otherwise the DevPaperFormat property is always overridden by the PageFormat property. You should always use PageFormat property to change the page format for the preview and the printer.

    In case of an error, LastError is set to VERR_COMMON. An error may occur if the device does not support changing the paper size at all, or not to the specified format.

Example:
    ActiveX / VCL
    
    Doc.DevPaperFormat = VPAPER_CSHEET  // C Sheet, 17- by 22-inches

    .NET
    

ActiveX / VCL: DevPaperFormat can be one of the following values

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPAPER_USER_DEFINED</td>
<td>0</td>
<td>User-Defined</td>
</tr>
<tr>
<td>VPAPER_A4</td>
<td>-1</td>
<td>A4 Sheet, 210- by 297-millimeters</td>
</tr>
<tr>
<td>VPAPER_LETTER</td>
<td>-2</td>
<td>US Letter, 8 1/2- by 11-inches</td>
</tr>
<tr>
<td>VPAPERLEGAL</td>
<td>-3</td>
<td>Legal, 8 1/2- by 14-inches</td>
</tr>
<tr>
<td>VPAPER_CSHEET</td>
<td>-4</td>
<td>C Sheet, 17- by 22-inches</td>
</tr>
<tr>
<td>VPAPER_DSheet</td>
<td>-5</td>
<td>D Sheet, 22- by 34-inches</td>
</tr>
<tr>
<td>VPAPER_ESHEET</td>
<td>-6</td>
<td>E Sheet, 34- by 44-inches</td>
</tr>
<tr>
<td>VPAPER_LETTERSMALL</td>
<td>-7</td>
<td>Letter Small, 8 1/2- by 11-inches</td>
</tr>
<tr>
<td>VPAPER_TABLOID</td>
<td>-8</td>
<td>Tabloid, 11- by 17-inches</td>
</tr>
<tr>
<td>VPAPER_LEDGER</td>
<td>-9</td>
<td>Ledger, 17- by 11-inches</td>
</tr>
<tr>
<td>VPAPER_STATEMENT</td>
<td>-10</td>
<td>Statement, 5 1/2- by 8 1/2-inches</td>
</tr>
<tr>
<td>VPAPER_EXECUTIVE</td>
<td>-11</td>
<td>Executive, 7 1/4- by 10 1/2-inches</td>
</tr>
<tr>
<td>VPAPER_A3</td>
<td>-12</td>
<td>A3 sheet, 297- by 420-millimeters</td>
</tr>
<tr>
<td>VPAPER_A4SMALL</td>
<td>-13</td>
<td>A4 small sheet, 210- by 297-millimeters</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>VPAPER_A5</td>
<td>-14</td>
<td>A5 sheet, 148- by 210-millimeters</td>
</tr>
<tr>
<td>VPAPER_B4</td>
<td>-15</td>
<td>B4 sheet, 250- by 354-millimeters</td>
</tr>
<tr>
<td>VPAPER_B5</td>
<td>-16</td>
<td>B5 sheet, 182- by 257-millimeter paper</td>
</tr>
<tr>
<td>VPAPER_FOLIO</td>
<td>-17</td>
<td>Folio, 8 1/2- by 13-inch paper</td>
</tr>
<tr>
<td>VPAPER_QUARTO</td>
<td>-18</td>
<td>Quarto, 215- by 275-millimeter paper</td>
</tr>
<tr>
<td>VPAPER_10X14</td>
<td>-19</td>
<td>10- by 14-inch sheet</td>
</tr>
<tr>
<td>VPAPER_11X17</td>
<td>-20</td>
<td>11- by 17-inch sheet</td>
</tr>
<tr>
<td>VPAPER_NOTE</td>
<td>-21</td>
<td>Note, 8 1/2- by 11-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_9</td>
<td>-22</td>
<td>#9 Envelope, 3 7/8- by 8 7/8-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_10</td>
<td>-23</td>
<td>#10 Envelope, 4 1/8- by 9 1/2-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_11</td>
<td>-24</td>
<td>#11 Envelope, 4 1/2- by 10 3/8-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_12</td>
<td>-25</td>
<td>#12 Envelope, 4 3/4- by 11-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_14</td>
<td>-26</td>
<td>#14 Envelope, 5- by 11 1/2-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_DL</td>
<td>-27</td>
<td>DL Envelope, 110- by 220-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_C5</td>
<td>-28</td>
<td>C5 Envelope, 162- by 229-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_C3</td>
<td>-29</td>
<td>C3 Envelope, 324- by 458-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_C4</td>
<td>-30</td>
<td>C4 Envelope, 229- by 324-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_C6</td>
<td>-31</td>
<td>C6 Envelope, 114- by 162-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_C65</td>
<td>-32</td>
<td>C65 Envelope, 114- by 229-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_B4</td>
<td>-33</td>
<td>B4 Envelope, 250- by 353-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_B5</td>
<td>-34</td>
<td>B5 Envelope, 176- by 250-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_B6</td>
<td>-35</td>
<td>B6 Envelope, 176- by 125-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_ITALY</td>
<td>-36</td>
<td>Italy Envelope, 110- by 230-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_MONARCH</td>
<td>-37</td>
<td>Monarch Envelope, 3 7/8- by 7 ½-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_PERSONAL</td>
<td>-38</td>
<td>6 3/4 Envelope, 3 5/8- by 6 1/2-inches</td>
</tr>
<tr>
<td>VPAPER_FANFOLD_US</td>
<td>-39</td>
<td>US Std Fanfold, 14 7/8- by 11-inches</td>
</tr>
<tr>
<td>VPAPER_FANFOLD_STD_GERMAN</td>
<td>-40</td>
<td>German Std Fanfold, 8 1/2- by 12-inches</td>
</tr>
<tr>
<td>VPAPER_FANFOLD_LGL_GERMAN</td>
<td>-41</td>
<td>German Legal Fanfold, 8 1/2- by 13-inches</td>
</tr>
<tr>
<td>VPAPER_ISO_B4</td>
<td>-42</td>
<td>B4 (ISO) 250 x 353 mm</td>
</tr>
<tr>
<td>VPAPER_JAPANESE_POSTCARD</td>
<td>-43</td>
<td>Japanese Postcard 100 x 148 mm</td>
</tr>
<tr>
<td>VPAPER_9X11</td>
<td>-44</td>
<td>9 x 11 in</td>
</tr>
<tr>
<td>VPAPER_10X11</td>
<td>-45</td>
<td>10 x 11 in</td>
</tr>
<tr>
<td>VPAPER_15X11</td>
<td>-46</td>
<td>15 x 11 in</td>
</tr>
<tr>
<td>VPAPER_ENV_INVITE</td>
<td>-47</td>
<td>Envelope Invite 220 x 220 mm</td>
</tr>
<tr>
<td>VPAPER_RESERVED_48</td>
<td>-48</td>
<td>RESERVED--DO NOT USE</td>
</tr>
<tr>
<td>VPAPER_RESERVED_49</td>
<td>-49</td>
<td>RESERVED--DO NOT USE</td>
</tr>
<tr>
<td>VPAPER_LETTER_EXTRA</td>
<td>-50</td>
<td>Letter Extra 9 2/75 x 12 in</td>
</tr>
<tr>
<td>VPAPER_LEGAL_EXTRA</td>
<td>-51</td>
<td>Legal Extra 9 2/75 x 15 in</td>
</tr>
<tr>
<td>VPAPER_TABLOID_EXTRA</td>
<td>-52</td>
<td>Tabloid Extra 11.69 x 18 in</td>
</tr>
<tr>
<td>VPAPER_A4_EXTRA</td>
<td>-53</td>
<td>A4 Extra 9.27 x 12.69 in</td>
</tr>
<tr>
<td>VPAPER_LETTER_TRANSVERSE</td>
<td>-54</td>
<td>Letter Transverse 8 2/75 x 11 in</td>
</tr>
<tr>
<td>VPAPER_A4_TRANSVERSE</td>
<td>-55</td>
<td>A4 Transverse 210 x 297 mm</td>
</tr>
<tr>
<td>VPAPER_LETTER_EXTRA_TRANSVERSE</td>
<td>-56</td>
<td>Letter Extra Transverse 9 2/75 x 12 in</td>
</tr>
<tr>
<td>VPAPER_A_PLUS</td>
<td>-57</td>
<td>SuperA/SuperA/A4 227 x 356 mm</td>
</tr>
<tr>
<td>VPAPER_B_PLUS</td>
<td>-58</td>
<td>SuperB/SuperB/A3 305 x 487 mm</td>
</tr>
<tr>
<td>VPAPER_LETTER_PLUS</td>
<td>-59</td>
<td>Letter Plus 8.5 x 12.69 in</td>
</tr>
<tr>
<td>VPAPER_A4_PLUS</td>
<td>-60</td>
<td>A4 Plus 210 x 330 mm</td>
</tr>
<tr>
<td>Device Control Properties VPE Control Reference</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPAPER_A5_TRANSVERSE -61 A5 Transverse 148 x 210 mm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPAPER_B5_TRANSVERSE -62 B5 (JIS) Transverse 182 x 257 mm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPAPER_A3_EXTRA -63 A3 Extra 322 x 445 mm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPAPER_A5_EXTRA -64 A5 Extra 174 x 235 mm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPAPER_B5_EXTRA -65 B5 (ISO) Extra 201 x 276 mm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPAPER_A2 -66 A2 420 x 594 mm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPAPER_A3_TRANSVERSE -67 A3 Transverse 297 x 420 mm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPAPER_A3_EXTRA_TRANSVERSE -68 A3 Extra Transverse 322 x 445 mm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Windows 2000 or Higher:**

| VPAPER_DBL_JAPANESE_POSTCARD -69 Japanese Double Postcard 200 x 148 mm |
| VPAPER_A6 -70 A6 105 x 148 mm |
| VPAPER_JENV_KAKU2 -71 Japanese Envelope Kaku #2 |
| VPAPER_JENV_KAKU3 -72 Japanese Envelope Kaku #3 |
| VPAPER_JENV_CHOU3 -73 Japanese Envelope Chou #3 |
| VPAPER_JENV_CHOU4 -74 Japanese Envelope Chou #4 |
| VPAPER_LETTER_ROTATED -75 Letter Rotated 11 x 8 1/2 in |
| VPAPER_A3_ROTATED -76 A3 Rotated 420 x 297 mm |
| VPAPER_A4_ROTATED -77 A4 Rotated 297 x 210 mm |
| VPAPER_A5_ROTATED -78 A5 Rotated 210 x 148 mm |
| VPAPER_B4_JIS_ROTATED -79 B4 (JIS) Rotated 364 x 257 mm |
| VPAPER_B5_JIS_ROTATED -80 B5 (JIS) Rotated 257 x 182 mm |
| VPAPER_JAPANESE_POSTCARD_ROTATED -81 Japanese Postcard Rotated 148 x 100 mm |
| VPAPER_DBL_JAPANESE_POSTCARD_ROTATED -82 Double Japanese Postcard Rotated 148 x 200mm |
| VPAPER_A6_ROTATED -83 A6 Rotated 148 x 105 mm |
| VPAPER_JENV_KAKU2_ROTATED -84 Japanese Envelope Kaku #2 Rotated |
| VPAPER_JENV_KAKU3_ROTATED -85 Japanese Envelope Kaku #3 Rotated |
| VPAPER_JENV_CHOU3_ROTATED -86 Japanese Envelope Chou #3 Rotated |
| VPAPER_JENV_CHOU4_ROTATED -87 Japanese Envelope Chou #4 Rotated |
| VPAPER_B6_JIS -88 B6 (JIS) 128 x 182 mm |
| VPAPER_B6_JIS_ROTATED -89 B6 (JIS) Rotated 182 x 128 mm |
| VPAPER_12X11 -90 12 x 11 in |
| VPAPER_JENV_YOU4 -91 Japanese Envelope You #4 |
| VPAPER_JENV_YOU4_ROTATED -92 Japanese Envelope You #4 Rotated |
| VPAPER_P16K -93 PRC 16K 146 x 215 mm |
| VPAPER_P32K -94 PRC 32K 97 x 151 mm |
| VPAPER_P32KBIG -95 PRC 32K(Big) 97 x 151 mm |
| VPAPER_PENV_1 -96 PRC Envelope #1 102 x 165 mm |
| VPAPER_PENV_2 -97 PRC Envelope #2 102 x 176 mm |
| VPAPER_PENV_3 -98 PRC Envelope #3 125 x 176 mm |
| VPAPER_PENV_4 -99 PRC Envelope #4 110 x 208 mm |
| VPAPER_PENV_5 -100 PRC Envelope #5 110 x 220 mm |
| VPAPER_PENV_6 -101 PRC Envelope #6 120 x 230 mm |
| VPAPER_PENV_7 -102 PRC Envelope #7 160 x 230 mm |
| VPAPER_PENV_8 -103 PRC Envelope #8 120 x 309 mm |
| VPAPER_PENV_9 -104 PRC Envelope #9 229 x 324 mm |
| VPAPER_PENV_10 -105 PRC Envelope #10 324 x 458 mm |
| VPAPER_P16K_ROTATED -106 PRC 16K Rotated |
| VPAPER_P32K_ROTATED -107 PRC 32K Rotated |
• VPAPER_P32KBIG_ROTATED -108 PRC 32K(Big) Rotated
• VPAPER_PENV_1_ROTATED -109 PRC Envelope #1 Rotated 165 x 102 mm
• VPAPER_PENV_2_ROTATED -110 PRC Envelope #2 Rotated 176 x 102 mm
• VPAPER_PENV_3_ROTATED -111 PRC Envelope #3 Rotated 176 x 125 mm
• VPAPER_PENV_4_ROTATED -112 PRC Envelope #4 Rotated 208 x 110 mm
• VPAPER_PENV_5_ROTATED -113 PRC Envelope #5 Rotated 220 x 110 mm
• VPAPER_PENV_6_ROTATED -114 PRC Envelope #6 Rotated 230 x 120 mm
• VPAPER_PENV_7_ROTATED -115 PRC Envelope #7 Rotated 230 x 160 mm
• VPAPER_PENV_8_ROTATED -116 PRC Envelope #8 Rotated 309 x 120 mm
• VPAPER_PENV_9_ROTATED -117 PRC Envelope #9 Rotated 324 x 229 mm
• VPAPER_PENV_10_ROTATED -118 PRC Envelope #10 Rotated 458 x 324 mm

.NET, Java, PHP, Python, Ruby, etc.: **DevPaperFormat** can be one of the following values

```java
public enum PageFormat {
    A4, A4 Sheet, 210- by 297-millimeters
    Letter, US Letter, 8 1/2- by 11-inches
    Legal, Legal, 8 1/2- by 14-inches
    CSheet, C Sheet, 17- by 22-inches
    DSheet, D Sheet, 22- by 34-inches
    ESheet, E Sheet, 34- by 44-inches
    LetterSmall, Letter Small, 8 1/2- by 11-inches
    Tabloid, Tabloid, 11- by 17-inches
    Ledger, Ledger, 17- by 11-inches
    Statement, Statement, 5 1/2- by 8 1/2-inches
    Executive, Executive, 7 1/4- by 10 1/2-inches
    A3, A3 sheet, 297- by 420-millimeters
    A4Small, A4 small sheet, 210- by 297-millimeters
    A5, A5 sheet, 148- by 210-millimeters
    B4, B4 sheet, 250- by 354-millimeters
    B5, B5 sheet, 182- by 257-millimeter paper
    Folio, Folio, 8 1/2- by 13-inch paper
    Quarto, Quarto, 215- by 275-millimeter paper
    Standard10x14, 10- by 14-inch sheet
    Standard11x17, 11- by 17-inch sheet
    Note, Note, 8 1/2- by 11-inches
    Envelope9, #9 Envelope, 3 7/8- by 8 7/8-inches
    Envelope10, #10 Envelope, 4 1/8- by 9 1/2-inches
    Envelope11, #11 Envelope, 4 1/2- by 10 3/8-inches
    Envelope12, #12 Envelope, 4 3/4- by 11-inches
    Envelope14, #14 Envelope, 5- by 11 1/2-inches
    EnvelopeDL, DL Envelope, 110- by 220-millimeters
    EnvelopeC5, C5 Envelope, 162- by 229-millimeters
    EnvelopeC3, C3 Envelope, 324- by 458-millimeters
    EnvelopeC4, C4 Envelope, 229- by 324-millimeters
    EnvelopeC6, C6 Envelope, 114- by 162-millimeters
    EnvelopeC65, C65 Envelope, 114- by 229-millimeters
    EnvelopeB4, B4 Envelope, 250- by 353-millimeters
```
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnvelopeB5</td>
<td>B5 Envelope, 176- by 250-millimeters</td>
</tr>
<tr>
<td>EnvelopeB6</td>
<td>B6 Envelope, 176- by 125-millimeters</td>
</tr>
<tr>
<td>EnvelopeItaly</td>
<td>Italy Envelope, 110- by 230-millimeters</td>
</tr>
<tr>
<td>EnvelopeMonarch</td>
<td>Monarch Envelope, 3 7/8- by 7 1/2-inches</td>
</tr>
<tr>
<td>EnvelopePersonal</td>
<td>6 3/4 Envelope, 3 5/8- by 6 1/2-inches</td>
</tr>
<tr>
<td>FanfoldUS</td>
<td>US Std Fanfold, 14 7/8- by 11-inches</td>
</tr>
<tr>
<td>FanfoldStdGerman</td>
<td>German Std Fanfold, 8 1/2- by 12-inches</td>
</tr>
<tr>
<td>FanfoldLglGerman</td>
<td>German Legal Fanfold, 8 1/2- by 13-inches</td>
</tr>
<tr>
<td>UserDefined</td>
<td>User Defined</td>
</tr>
<tr>
<td>IsoB4</td>
<td>B4 (ISO) 250 x 353 mm</td>
</tr>
<tr>
<td>JapanesePostcard</td>
<td>Japanese Postcard 100 x 148 mm</td>
</tr>
<tr>
<td>Standard9x11</td>
<td>9 x 11 in</td>
</tr>
<tr>
<td>Standard10x11</td>
<td>10 x 11 in</td>
</tr>
<tr>
<td>Standard15x11</td>
<td>15 x 11 in</td>
</tr>
<tr>
<td>EnvelopeInvite</td>
<td>Envelope Invite 220 x 220 mm</td>
</tr>
<tr>
<td>Reserved48</td>
<td>RESERVED--DO NOT USE</td>
</tr>
<tr>
<td>Reserved49</td>
<td>RESERVED--DO NOT USE</td>
</tr>
<tr>
<td>LetterExtra</td>
<td>Letter Extra 9 275 x 12 in</td>
</tr>
<tr>
<td>LegalExtra</td>
<td>Legal Extra 9 275 x 15 in</td>
</tr>
<tr>
<td>TabloidExtra</td>
<td>Tabloid Extra 11.69 x 18 in</td>
</tr>
<tr>
<td>A4Extra</td>
<td>A4 Extra 9.27 x 12.69 in</td>
</tr>
<tr>
<td>LetterTransverse</td>
<td>Letter Transverse 8 275 x 11 in</td>
</tr>
<tr>
<td>A4Transverse</td>
<td>A4 Transverse 210 x 297 mm</td>
</tr>
<tr>
<td>LetterExtraTransverse</td>
<td>Letter Extra Transverse 9 275 x 12 in</td>
</tr>
<tr>
<td>APlus</td>
<td>SuperA/SuperA/A4 227 x 356 mm</td>
</tr>
<tr>
<td>BPlus</td>
<td>SuperB/SuperB/A3 305 x 487 mm</td>
</tr>
<tr>
<td>LetterPlus</td>
<td>Letter Plus 8.5 x 12.69 in</td>
</tr>
<tr>
<td>A4Plus</td>
<td>A4 Plus 210 x 330 mm</td>
</tr>
<tr>
<td>A5Transverse</td>
<td>A5 Transverse 148 x 210 mm</td>
</tr>
<tr>
<td>B5Transverse</td>
<td>B5 (JIS) Transverse 182 x 257 mm</td>
</tr>
<tr>
<td>A3Extra</td>
<td>A3 Extra 322 x 445 mm</td>
</tr>
<tr>
<td>A5Extra</td>
<td>A5 Extra 174 x 235 mm</td>
</tr>
<tr>
<td>B5Extra</td>
<td>B5 (ISO) Extra 201 x 276 mm</td>
</tr>
<tr>
<td>A2</td>
<td>A2.420 x 594 mm</td>
</tr>
<tr>
<td>A3Transverse</td>
<td>A3 Transverse 297 x 420 mm</td>
</tr>
<tr>
<td>A3ExtraTransverse</td>
<td>A3 Extra Transverse 322 x 445 mm</td>
</tr>
</tbody>
</table>

**Windows 2000 or Higher:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DblJapanesePostcard</td>
<td>Japanese Double Postcard 200 x 148 mm</td>
</tr>
<tr>
<td>A6</td>
<td>A6 105 x 148 mm</td>
</tr>
<tr>
<td>JapaneseEnvelopeKaku2</td>
<td>Japanese Envelope Kaku #2</td>
</tr>
<tr>
<td>JapaneseEnvelopeKaku3</td>
<td>Japanese Envelope Kaku #3</td>
</tr>
<tr>
<td>JapaneseEnvelopeChou3</td>
<td>Japanese Envelope Chou #3</td>
</tr>
<tr>
<td>JapaneseEnvelopeChou4</td>
<td>Japanese Envelope Chou #4</td>
</tr>
<tr>
<td>LetterRotated</td>
<td>Letter Rotated 11 x 8 1/2 in</td>
</tr>
<tr>
<td>A3Rotated</td>
<td>A3 Rotated 420 x 297 mm</td>
</tr>
<tr>
<td>A4Rotated</td>
<td>A4 Rotated 297 x 210 mm</td>
</tr>
<tr>
<td>A5Rotated</td>
<td>A5 Rotated 210 x 148 mm</td>
</tr>
<tr>
<td>B4JisRotated</td>
<td>B4 (JIS) Rotated 364 x 257 mm</td>
</tr>
<tr>
<td>B5JisRotated</td>
<td>B5 (JIS) Rotated 257 x 182 mm</td>
</tr>
</tbody>
</table>
JapanesePostcardRotated,  Japanese Postcard Rotated 148 x 100 mm
DbJapanesePostcardRotated, Double Japanese Postcard Rotated 148 x 200 mm
A6Rotated,  A6 Rotated 148 x 105 mm
JapaneseEnvelopeKaku2Rotated, Japanese Envelope Kaku #2 Rotated
JapaneseEnvelopeKaku3Rotated, Japanese Envelope Kaku #3 Rotated
JapaneseEnvelopeChou3Rotated, Japanese Envelope Chou #3 Rotated
JapaneseEnvelopeChou4Rotated, Japanese Envelope Chou #4 Rotated
B6Jis,  B6 (JIS) 128 x 182 mm
B6JisRotated,  B6 (JIS) Rotated 182 x 128 mm
Standard12x11,  12 x 11 in
JapaneseEnvelopeYou4, Japanese Envelope You #4
JapaneseEnvelopeYou4Rotated, Japanese Envelope You #4 Rotated
Prc16K,  PRC 16K 146 x 215 mm
Prc32K,  PRC 32K 97 x 151 mm
Prc32KBig,  PRC 32K(Big) 97 x 151 mm
PrcEnvelope1, PRC Envelope #1 102 x 165 mm
PrcEnvelope2, PRC Envelope #2 102 x 176 mm
PrcEnvelope3, PRC Envelope #3 125 x 176 mm
PrcEnvelope4, PRC Envelope #4 110 x 208 mm
PrcEnvelope5, PRC Envelope #5 110 x 220 mm
PrcEnvelope6, PRC Envelope #6 120 x 230 mm
PrcEnvelope7, PRC Envelope #7 160 x 230 mm
PrcEnvelope8, PRC Envelope #8 120 x 309 mm
PrcEnvelope9, PRC Envelope #9 229 x 324 mm
PrcEnvelope10, PRC Envelope #10 324 x 458 mm
Prc16KRotated, PRC 16K Rotated
Prc32KRotated, PRC 32K Rotated
Prc32KBigRotated, PRC 32K(Big) Rotated
PrcEnvelope1Rotated, PRC Envelope #1 Rotated 165 x 102 mm
PrcEnvelope2Rotated, PRC Envelope #2 Rotated 176 x 102 mm
PrcEnvelope3Rotated, PRC Envelope #3 Rotated 176 x 125 mm
PrcEnvelope4Rotated, PRC Envelope #4 Rotated 208 x 110 mm
PrcEnvelope5Rotated, PRC Envelope #5 Rotated 220 x 110 mm
PrcEnvelope6Rotated, PRC Envelope #6 Rotated 230 x 120 mm
PrcEnvelope7Rotated, PRC Envelope #7 Rotated 230 x 160 mm
PrcEnvelope8Rotated, PRC Envelope #8 Rotated 309 x 120 mm
PrcEnvelope9Rotated, PRC Envelope #9 Rotated 324 x 229 mm
PrcEnvelope10Rotated, PRC Envelope #10 Rotated 458 x 324 mm
8.7 DevPaperWidth

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets an individual paper width for the currently selected output device to the specified value. Setting this property does NOT affect the setting of the preview. If you set the width, you should also set the height. If this call is successful, DevPaperFormat returns VPAPER_USER_DEFINED (=0).

property VpeCoord VPE.DevPaperWidth

read / write; runtime only

Possible Values:
the paper width

Remarks:
In case of an error, LastError is set to VERR_COMMON. An error may occur if the device does not support changing the paper width.

This property is available for total control. It is only useful if called while processing the PRINT_NEWPAGE event. Otherwise this property is always overridden by the PageFormat or PageWidth property.

We experienced that this option doesn’t work with some printer drivers. We tested this for example on an Epson LQ-550 dot-matrix printer and it didn't work on WiW 3.11 and NT 3.51, but it worked on Win95. But we heared about, that a LQ-510 printer driver will solve the problem for the LQ-550. So if your printer does not respond to this setting, it is a printer driver problem. In that case you can try to use another compatible printer driver, or see if the printer driver responds to setting the paper size with the property DevPaperFormat to a predefined paper format.

Example:

```
Doc.DevPaperWidth = 10.5 // 10.5 cm / inch
```
8.8 DevPaperHeight

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets an individual paper height for the currently selected output device to the specified value. Setting this property does NOT affect the setting of the preview. If you set the height, you should also set the width. If this call is successful, DevPaperFormat returns VPAPER_USER_DEFINED (=0). Setting this property makes sense for tractor printers with endless paper feeders.

**property long VPE.DevPaperHeight**

read / write; runtime only

**Possible Values:**

the paper height

**Remarks:**

In case of an error, LastError is set to VERR_COMMON. An error may occur if the device does not support changing the paper height.

This property is available for total control. It is only useful if called while processing the PRINT_NEWPAGE event. Otherwise this property is always overridden by the PageFormat or PageHeight property.

We experienced that this option doesn’t work with some printer drivers. We tested this for example on an Epson LQ-550 dot-matrix printer and it didn’t work on WfW 3.11 and NT 3.51, but it worked on Win95. But we heared about, that a LQ-510 printer driver will solve the problem for the LQ-550. So if your printer does not respond to this setting, it is a printer driver problem. In that case you can try to use another compatible printer driver, or see if the printer driver responds to setting the paper size with the property DevPaperFormat to a predefined paper format.

**Example:**

```
Doc.DevPaperHeight = 5.8 // 5.8 cm / inch
```
8.9 DevScalePercent

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets the printing scale of the currently selected output device to the specified value. Setting this property does NOT affect the setting of the preview. The value you specify is in percent.

property integer VPE.DevScalePercent

read / write; runtime only

Possible Values:
the scaling (e.g. 100 = 100%; 25 = 25%)

Remarks:
In case of an error, LastError is set to VERR_COMMON. An error may occur if the device does not support changing the scale.
8.10 DevPrintQuality

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets the print quality of the currently selected output device to the specified value or it sets the x-resolution in DPI.

**property DevPrintQuality [integer] VPE.DevPrintQuality**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VRES_DRAFT</td>
<td>-1</td>
<td>Draft</td>
<td>Draft Quality</td>
</tr>
<tr>
<td>VRES_LOW</td>
<td>-2</td>
<td>Low</td>
<td>Low Quality</td>
</tr>
<tr>
<td>VRES_MEDIUM</td>
<td>-3</td>
<td>Medium</td>
<td>Medium Quality</td>
</tr>
<tr>
<td>VRES_HIGH</td>
<td>-4</td>
<td>High</td>
<td>High Quality</td>
</tr>
</tbody>
</table>

If a positive value is given, it specifies the number of dots per inch (DPI) for the x-resolution and is therefore device dependent. If you are able to set `DevYResolution` without error, this property should specify the x-resolution in DPI.

**Remarks:**

In case of an error, `LastError` is set to VERR_COMMON. An error may occur if the device does not support setting the print quality or x-resolution. You should read this property’s value after setting it, to be sure the value has been accepted. Sometimes the value has not been accepted, but `LastError` returns no error state (= VERR_OK).

We experienced that some drivers do not allow setting the y-resolution, BEFORE the x-resolution had been changed and vice versa. Also some drivers only accept the same values for both resolutions.

**Example:**

**ActiveX / VCL:**

```vcl
Doc.DevPrintQuality = VRES_DRAFT   // Draft Mode
Doc.DevPrintQuality = 300         // 300 DPI
```

**.NET:**

```csharp
Doc.DevPrintQuality = (DevPrintQuality)300   // 300 DPI
```
8.11 DevYResolution

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets the y-resolution in dots per inch for the currently selected output device. If the output device initializes this property, the DevPrintQuality property specifies the x-resolution, in dots per inch (DPI), of the printer.

property integer VPE.DevYResolution

    read / write; runtime only

Possible Values:
    the y-resolution in DPI

Remarks:
    In case of an error, LastError is set to VERR_COMMON. An error may occur if the device does not support setting the y-resolution. You should read this property’s value after setting it, to be sure the value has been accepted. Sometimes the value has not been accepted, but LastError returns no error state (= VERR_OK).

    We experienced that some drivers do not allow setting the y-resolution, BEFORE the x-resolution had been changed and vice versa. Also some drivers only accept the same values for both resolutions.

Example:

    Doc.DevPrintQuality = 300  // 300 DPI X-Resolution
    Doc.DevYResolution = 300  // 300 DPI Y-Resolution
8.12 DevColor

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets the color mode for the currently selected output device, only useful, if the output device is a color printer.

**property DevColor [integer] VPE.DevColor**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCOLOR_MONOCHROME</td>
<td>1</td>
<td>Monochrome</td>
<td></td>
</tr>
<tr>
<td>VCOLOR_COLOR</td>
<td>2</td>
<td>Color</td>
<td></td>
</tr>
</tbody>
</table>

**Remarks:**

In case of an error, [LastError](#) is set to VERR_COMMON. An error may occur if the device does not support setting the color mode.

**Example:**

**ActiveX / VCL**

```csharp
Doc.DevColor = VCOLOR_MONOCHROME
```

**.NET**

```csharp
Doc.DevColor = DevColor.Monochrome
```
8.13 DevDuplex

[Windows platform only; not supported by PHP and the Community Edition]

Selects duplex (or double-sided) printing for the currently selected output device (if it is capable of duplex printing).

**property DevDuplex [integer] VPE.DevDuplex**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDUP_SIMPLEX</td>
<td>1</td>
<td>Simplex</td>
<td></td>
</tr>
<tr>
<td>VDUP_VERTICAL</td>
<td>2</td>
<td>Vertical</td>
<td></td>
</tr>
<tr>
<td>VDUP_HORIZONTAL</td>
<td>3</td>
<td>Horizontal</td>
<td></td>
</tr>
</tbody>
</table>

**Remarks:**
In case of an error, LastError is set to VERR_COMMON. An error may occur if the device does not support setting the duplex mode.

**Example:**

ActiveX / VCL

```plaintext
Doc.DevDuplex = VDUP_VERTICAL
```

.NET

```plaintext
Doc.DevDuplex = DevDuplex.Vertical
```
8.14 DevTTOption

[Windows platform only; not supported by PHP and the Community Edition]

Returns / specifies how TrueType® fonts should be printed on the currently selected output device.

property DevTTOption [integer] VPE.DevTTOption

read / write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VTT_BITMAP</td>
<td>1</td>
<td>Bitmap</td>
<td>Prints TrueType fonts as graphics. This is the default action for dot-matrix printers.</td>
</tr>
<tr>
<td>VTT_DOWNLOAD</td>
<td>2</td>
<td>Download</td>
<td>Downloads TrueType fonts as soft fonts. This is the default action for Hewlett-Packard printers that use Printer Control Language (PCL).</td>
</tr>
<tr>
<td>VTT_SUBDEV</td>
<td>3</td>
<td>Subdev</td>
<td>Substitute device fonts for TrueType fonts. This is the default action for PostScript® printers.</td>
</tr>
</tbody>
</table>

Remarks:

In case of an error, LastError is set to VERR_COMMON. An error may occur if the device does not support setting the TTOption. Setting this property doesn't work with some printer drivers. For example it doesn't work with our HP4 and HP5 printer drivers on WfW 3.11 and NT 3.51, but it works with both drivers on Win95. This is a driver problem.

Example:

ActiveX / VCL

```
Doc.DevTTOption = VTT_BITMAP
```

.NET

```
Doc.DevTTOption = DevTTOption.Bitmap
```

The following note is from Microsoft itself (MSDN column "Ask Dr. Gui")

These options are usually available in the printer properties dialog box under the Fonts, Device Options, or Advanced Settings tabs. Printer drivers for HP printers have an option called "Text As Graphics." This option, if turned on, prevents use of device fonts and draws the text using the operating system version of the font. PostScript printer drivers sometimes have options that are set per font and they usually have options to only download the font rather than rasterize it. Whenever these options are selected, the print job will get larger because the rasterized glyphs of the font are included within it.

It should save you lots of time and frustration to note that printer drivers are more different than alike, and that these settings are private settings for each printer. It may be necessary to explore the printer's settings to find the one that does the trick.

In theory, how a printer driver works with a TrueType font on the device is controllable by the member of a DEVMODE structure. In practice, however, Dr. GUI has diagnosed plenty of ill printer drivers that do not use this member of the DEVMODE structure correctly. This is a pity, because it places the burden of configuring the printer to use the operating system's fonts on the shoulders of an application's users.
8.15 DevEnumPaperBins

[Windows platform only; not supported by PHP and the Community Edition]

Initialize enumeration of all available paper bins for the currently selected output device. After this function has been called, you can retrieve each single Paper Bin name with the method GetDevPaperBinName().

method long VPE.DevEnumPaperBins(
)

Returns:
the number of available paper bins of the currently selected output device

Remarks:
In case of an error, LastError is set to VERR_COMMON. An error may occur if the device does not support multiple paper bins.

Do not call DevEnumPaperBins() in a loop like "for x = 0 to DevEnumPaperBins() - 1", because each time VPE will check for all available devices, which is time-consuming. Rather assign the value of DevEnumPaperBins() to a variable.

Not all of the bin options are available on every printer. Check the printer documentation for more specific descriptions of these options.

Example:
see GetDevPaperBinName()
8.16 GetDevPaperBinName

[Windows platform only; not supported by PHP and the Community Edition]

Returns the paper bin name specified by "index" of the currently selected output device. If index is out of range, an empty string (""") will be returned. This method only works correctly if you called DevEnumPaperBins() before.

```csharp
method string VPE.GetDevPaperBinName(
    long Index
)
```

*long Index*

must be in the range between 0 and the value returned by DevEnumPaperBins() - 1

**Returns:**

the Paper Bin name

**Remarks:**

In case of an error, LastError is set to VERR_COMMON. This method only works correctly if you called DevEnumPaperBins() before.

**Example:**

```csharp
long count, i
string s
count = Doc.DevEnumPaperBins() - 1 // initialize enumeration
for i = 0 to count
    s = Doc.GetDevPaperBinName(i)
    MessageBox(s)  // show each available bin in a separate message box
next i
```

Do not call DevEnumPaperBins() repeatedly, because each time VPE will query all available paper bins of the currently selected output device from the system.

**WRONG EXAMPLE:**

```csharp
long i
string s
for i = 0 to Doc.DevEnumPaperBins() - 1
    s = Doc.GetDevPaperBinName(i)
    MessageBox(s)  // show each available bin in a separate message box
next i
```
8.17 GetDevPaperBinID

[Windows platform only; not supported by PHP and the Community Edition]

With this method you can retrieve all available paper bins of the currently selected output device. It returns the Bin-ID (see DevPaperBin) specified by "index" of the currently selected output device. The Bin-ID’s are numeric Windows system constants to identify a bin. This method only works correctly if you called DevEnumPaperBins() before.


  **long Index**

  )

*int index*
  
  must be in the range between 0 and the value returned by DevEnumPaperBins() - 1

**Returns:**
the Bin-ID, possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBIN_UNTOUCHED</td>
<td>0</td>
<td>Untouched</td>
<td></td>
</tr>
<tr>
<td>VBIN_UPPER</td>
<td>1</td>
<td>Upper</td>
<td></td>
</tr>
<tr>
<td>VBIN_ONLYONE</td>
<td>1</td>
<td>OnlyOne</td>
<td></td>
</tr>
<tr>
<td>VBIN_LOWER</td>
<td>2</td>
<td>Lower</td>
<td></td>
</tr>
<tr>
<td>VBIN_MIDDLE</td>
<td>3</td>
<td>Middle</td>
<td></td>
</tr>
<tr>
<td>VBIN_MANUAL</td>
<td>4</td>
<td>Manual</td>
<td></td>
</tr>
<tr>
<td>VBIN_ENVELOPE</td>
<td>5</td>
<td>Envelope</td>
<td></td>
</tr>
<tr>
<td>VBIN_ENVMANUAL</td>
<td>6</td>
<td>EnvelopeManual</td>
<td></td>
</tr>
<tr>
<td>VBIN_AUTO</td>
<td>7</td>
<td>Auto</td>
<td></td>
</tr>
<tr>
<td>VBIN TRACTOR</td>
<td>8</td>
<td>Tractor</td>
<td></td>
</tr>
<tr>
<td>VBIN_SMALLFMT</td>
<td>9</td>
<td>SmallFormat</td>
<td></td>
</tr>
<tr>
<td>VBIN_LARGEFMT</td>
<td>10</td>
<td>LargeFormat</td>
<td></td>
</tr>
<tr>
<td>VBIN_LARGECAPACITY</td>
<td>11</td>
<td>LargeCapacity</td>
<td></td>
</tr>
<tr>
<td>VBIN_CASSETTE</td>
<td>14</td>
<td>Cassette</td>
<td></td>
</tr>
</tbody>
</table>

Additionally, each printer driver may define its own constants for other bins. Not all of the bin options are available on every printer. Check the printer documentation for more specific descriptions of these options.

**Remarks:**
In case of an error, *LastError* is set to VERR_COMMON. This method works only correctly, if you called DevEnumPaperBins() before.

**Example:**
see DevPaperBin
8.18 DevPaperBin
[Windows platform only; not supported by PHP and the Community Edition]
Selects a paper bin for the currently selected output device.

**property PaperBin [long] VPE.DevPaperBin**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBIN_UPPER</td>
<td>1</td>
<td>Upper</td>
<td></td>
</tr>
<tr>
<td>VBIN_ONLYONE</td>
<td>1</td>
<td>OnlyOne</td>
<td></td>
</tr>
<tr>
<td>VBIN_LOWER</td>
<td>2</td>
<td>Lower</td>
<td></td>
</tr>
<tr>
<td>VBIN_MIDDLE</td>
<td>3</td>
<td>Middle</td>
<td></td>
</tr>
<tr>
<td>VBIN_MANUAL</td>
<td>4</td>
<td>Manual</td>
<td></td>
</tr>
<tr>
<td>VBIN_ENVELOPE</td>
<td>5</td>
<td>Envelope</td>
<td></td>
</tr>
<tr>
<td>VBIN_ENVMANUAL</td>
<td>6</td>
<td>EnvelopeManual</td>
<td></td>
</tr>
<tr>
<td>VBIN_AUTO</td>
<td>7</td>
<td>Auto</td>
<td></td>
</tr>
<tr>
<td>VBIN_TRACTOR</td>
<td>8</td>
<td>Tractor</td>
<td></td>
</tr>
<tr>
<td>VBIN_SMALLFMT</td>
<td>9</td>
<td>SmallFormat</td>
<td></td>
</tr>
<tr>
<td>VBIN_LARGEFMT</td>
<td>10</td>
<td>LargeFormat</td>
<td></td>
</tr>
<tr>
<td>VBIN_LARGECAPACITY</td>
<td>11</td>
<td>LargeCapacity</td>
<td></td>
</tr>
<tr>
<td>VBIN_CASSETTE</td>
<td>14</td>
<td>Cassette</td>
<td></td>
</tr>
</tbody>
</table>

Additionally, each printer driver may define its own constants for other bins (see Remarks).

Not all of the bin options are available on every printer. Check the printer documentation for more specific descriptions of these options.

**Remarks:**

In case of an error, LastError is set to VERR_COMMON. An error may occur if the device does not support multiple paper bins.

The Bin-ID’s are numeric Windows system constants to identify a bin. Additionally each printer driver may define its own constants for other bins. To retrieve a Bin-ID defined by a printer driver, enumerate all bins with DevEnumPaperBins() and retrieve each Bin-ID associated with a Bin-Entry with GetDevPaperBinID()

The DevPaperBin property is available for total control. It is only useful if called while processing the BeforePrintNewPage() event (VCL: OnPrintNewPage(), .NET: BeforePrintNewPage()). Otherwise the DevPaperBin property is always overridden by the PaperBin property, which is specified separately for each page. **You should always use the PaperBin property to change the bin for the printer.**

It has been reported that several printer drivers - even from respectable manufacturers - do not behave correctly. Our own tests revealed for example that the HP 2200 D driver can only switch once the bin, but thereafter it will not switch the bin again. The HP 5P driver behaves
correctly unless multiple copies and collation are selected. If collation is activated, the driver will print all pages except the first multiple times as if non-collation was selected.

Example:

```csharp
long count, i
string s
count = Doc.DevEnumPaperBins() - 1 // initialize enumeration
for i = 0 to count
    // retrieve the name of the i-th paper bin
    s = Doc.GetDevPaperBinName(i)

    // show each available paper bin name in a separate message box
    MessageBox(s)

    // select the bin
next i
```

or, to select a bin directly:

**ActiveX / VCL**

```csharp
Doc.DevPaperBin = VBIN_LOWER
```

**.NET**

```csharp
```
8.19 DevPrinterOffsetX

[Windows platform only; not supported by PHP and the Community Edition]

Returns the leftmost x-position where the printer can print on for the currently selected output device. Printers have an unprintable area on the paper. This is the area around the paper margins where a printer can not print on due to technical limitations of the printing mechanism.

**property VpeCoord VPE.DevPrinterOffsetX**

- read; runtime only

**Returns:**
- the leftmost x-position where the printer starts printing

**Remarks:**
- In case of an error, LastError is set to VERR_COMMON.
8.20 DevPrinterOffsetY

[Windows platform only; not supported by PHP and the Community Edition]

Returns the topmost y-position where the printer can print on for the currently selected output device. Printers have an unprintable area on the paper. This is the area around the paper margins where a printer can not print on due to technical limitations of the printing mechanism.

property VpeCoord VPE.DevPrinterOffsetY

   read; runtime only

Returns:
   the topmost y-position where the printer starts printing

Remarks:
   In case of an error, LastError is set to VERR_COMMON.
8.21 DevPrintableWidth

[Windows platform only; not supported by PHP and the Community Edition]

Returns the printable width for the currently selected output device. Printers have an unprintable area on the paper. This is the area around the paper margins where a printer can not print on due to technical limitations of the printing mechanism. The value returned by this function considers the unprintable area of the left and right margin and returns the resulting printable width.

**property VpeCoord VPE.DevPrintableWidth**

- **read; runtime only**

**Returns:**
- the printable width

**Remarks:**
- In case of an error, LastError is set to VERR_COMMON.
8.22 DevPrintableHeight

[Windows platform only; not supported by PHP and the Community Edition]

Returns the printable height for the currently selected output device. Printers have an unprintable area on the paper. This is the area around the paper margins where a printer can not print on due to technical limitations of the printing mechanism. The value returned by this function considers the unprintable area of the top and bottom margin and returns the resulting printable height.

property VpeCoord VPE.DevPrintableHeight

    read; runtime only

**Returns:**

    the printable height

**Remarks:**

    In case of an error, **LastError** is set to VERR_COMMON.
8.23 DevPhysPageWidth

[Windows platform only; not supported by PHP and the Community Edition]

Retrieves the total page width of the currently selected output device.

property VpeCoord VPE.DevPhysPageWidth

read; runtime only

Returns:

the total page width of the currently selected output device

Remarks:

In case of an error, LastError is set to VERR_COMMON.
8.24 DevPhysPageHeight  
[Windows platform only; not supported by PHP and the Community Edition]  
Retrieves the total page height of the currently selected output device.

**property VpeCoord VPE.DevPhysPageHeight**

read; runtime only

**Returns:**
the total page height of the currently selected output device

**Remarks:**
In case of an error, `LastError` is set to VERR_COMMON.
8.25 DevCopies

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets the number of copies for the currently selected output device.

**property long VPE.DevCopies**

read / write; runtime only

**Possible Values:**
the number of copies to print

**Remarks:**
It is possible to set the default number of copies for most printers in the printer control panel. This value is automatically reflected by DevCopies at the moment a device is selected (i.e. when calling OpenDoc() or when you set explicitly the property Device).
8.26 DevCollate

[Windows platform only; not supported by PHP and the Community Edition]

Returns / specifies, if printed copies shall be collated on the currently selected output device.

**property boolean VPE.DevCollate**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>collate</td>
</tr>
<tr>
<td>False</td>
<td>do not collate</td>
</tr>
</tbody>
</table>
8.27 DevFromPage

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets the starting page that shall be printed onto the currently selected output device.

property long VPE.DevFromPage

read / write; runtime only

Possible Values:
the starting page

Remarks:
This property is not written to a VPE setup file.
8.28 DevToPage

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets the ending page that shall be printed onto the currently selected output device.

**property long VPE.DevToPage**

read / write; runtime only

**Possible Values:**
the ending page

**Remarks:**
This property is not written to a VPE setup file.
8.29 DevToFile

[Windows platform only; not supported by PHP and the Community Edition]

Returns / specifies for the currently selected output device if the document shall be printed into a file.

**property boolean VPE.DevToFile**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>print to file</td>
</tr>
<tr>
<td>False</td>
<td>do not print to file</td>
</tr>
</tbody>
</table>

**Remarks:**

This property is not written to a VPE setup file.
8.30 DevFileName

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets for the currently selected output device the file name that is used, if the document is printed into a file (see: DevToFile). If this property is not set or reset to an empty string (""), a dialog box will pop-up when the print is started, asking the user for the file name.

With some printer drivers, this dialog box will not pop up.

**property string VPE.DevFileName**

  read / write; runtime only

**Possible Values:**

  the (path- and) filename the document is printed to

**Remarks:**

  This property is not written to a VPE setup file.
8.31 DevJobName

[Windows platform only; not supported by PHP and the Community Edition]

Returns / sets the job name for the currently selected output device, which will be shown in the printer spooler of the system. If no job name is set ("" or NULL), VPE will compose a job name from the name of the application calling VPE and the title of the document (or of the filename, if SwapFileName is set).

property string VPE.DevJobName

read / write; runtime only

Possible Values:

job name

Remarks:

This property is not written to a VPE setup file.
8.32 DevSendData

[Windows platform only; not supported by PHP and the Community Edition]

This method enables your application to send printer dependent Escape-Sequences (control sequences and binary data) directly to the printer. Because the data you send with this method is strictly printer dependent, you must know to what printer model you are printing.

With this method it is possible, to select for example an output paper bin by code (or whatever other functionality is provided by the connected printer).

```csharp
method long VPE.DevSendData(
    string Data
)
```

**string Data**

the string with the data

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success</td>
</tr>
<tr>
<td>False</td>
<td>failure</td>
</tr>
</tbody>
</table>

**Remarks:**

In case of an error, `LastError` is set to VERR_COMMON. An error may occur if the device does not support sending binary data.

**ActiveX:** You may only call this function while processing the event `DoPrintDevData()`.

**VCL:** You may only call this function while processing the event `OnPrintDevData()`.

**.NET:** You may only call this function while processing the event `PrintDevData()`.

In C# the values true and false are not compatible with the integer datatype. Use 1 for true and 0 for false.

If you call `DevSendData()` while not processing the event, the method does nothing.

**Example:**

Example in Visual Basic (while processing the DoPrintDevData() event):

**ActiveX:**

```vbnet
Const RevLandScape = "&l3O" ' PCL command to change the Paper orientation to Reverse Landscape.
PCL_Escape$ = Chr$(27) + RevLandScape
Doc.DevSendData(PCL_Escape$)
ResultingAction = PRINT_ACTION_CHANGE
```

**.NET:**

```csharp
const string RevLandScape = "&l3O" // PCL command to change the Paper orientation to Reverse Landscape.
PCL_Escape = Chr(27) + RevLandScape
Doc.DevSendData(PCL_Escape)
event.PrintResultingAction = PrintResultingAction.Change
```

The characters in the string are: "&" + the letter "l" + "3" + the letter "O"
8.33 WritePrinterSetup

[Windows platform only; not supported by PHP and the Community Edition]

Writes the current printer setup to file for persistent storage. This file is 100% identical to the file that is generated by SetupPrinter().

```csharp
method boolean VPE.WritePrinterSetup(
    string FileName
)
```

`string FileName`
- the name of the file, the setup is written to

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success</td>
</tr>
<tr>
<td>False</td>
<td>failure</td>
</tr>
</tbody>
</table>

**Remarks:**

In case of an error, `LastError` is set to VERR_COMMON.

For standardization, Printer Setup Files created with VPE should have the suffix ".PRS"

Win16 and Win32 printer setup files generated by VPE are NOT identically and therefore can NOT be created on one platform and be used on another platform. VPE detects such a situation and will set `LastError = VERR_COMMON`.

The setup files contain private device driver data. This has the big advantage, that special settings, options and features of a specific printer can be set and stored to file (for example some printers offer to select an output paper bin, which is not known by the Win API). Therefore, the use of the method SetupPrinter() has some advantages, as the settings a user may make in the printer specific setup dialogs are stored with the file.

But - because of the private driver data - if the user replaces the printer by another model (or manufacturer) or possibly if just the driver version is changed, it might be necessary to delete the setup file(s) and to newly create them.
8.34 ReadPrinterSetup

[Windows platform only; not supported by PHP and the Community Edition]

Reads a printer setup file that was previously written by SetupPrinter() or WritePrinterSetup() and uses the settings contained in the file.

**method boolean VPE.ReadPrinterSetup(**

  **string FileName**

**)

**string FileName**

the name of the file, the setup is read from

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success</td>
</tr>
<tr>
<td>False</td>
<td>failure</td>
</tr>
</tbody>
</table>

**Remarks:**

In case of an error, **LastError** is set to VERR_COMMON.

see **WritePrinterSetup()**
9 Layout Functions

9.1 Layout Functions

These functions offer you powerful ways of controlling the layout of the document, of dynamically placing and sizing objects and to define page boundaries for every individual page within the virtual document.

9.2 UnitTransformation

Specifies the coordinate system in which the API of VPE handles coordinates. The coordinate system of VPE can either be in centimeter or inch units, as well as the old (prior to v4.0) 0.1mm units.

Internally, VPE computes object positions and dimensions with a precision of 1/10.000 mm.

**property double VPE. UnitTransformation**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VUNIT_FACTOR_CM</td>
<td>100000</td>
<td>Centimeter</td>
<td>coordinates are handled in cm units by the VPE API</td>
</tr>
<tr>
<td>VUNIT_FACTOR_INCH</td>
<td>254000</td>
<td>Inch</td>
<td>coordinates are handled in inch units by the VPE API</td>
</tr>
<tr>
<td>VUNIT_FACTOR_Mm10</td>
<td>1000</td>
<td>Mm10</td>
<td>coordinates are handled in 1/10 of a mm by the VPE API (compatible to VPE &lt; v4.0)</td>
</tr>
</tbody>
</table>

**Remarks:**

Setting this property does not affect the ruler’s unit measurement of the preview. To change the units of the rulers, use `RulersMeasure`.

**Examples:**

**ActiveX / VCL:**

```vbnet
Doc.UnitTransformation = VUNIT_FACTOR_Mm10
```

**.NET Visual Basic:**

```vbnet
Doc.UnitTransformation = UnitFactor.Mm10
```

**.NET C#:**

```vbnet
Doc.UnitTransformation = (double)UnitFactor.Mm10
```

Sets the unit transformation to the old 0.1 mm units.
9.3 EngineRenderMode

VPE v4.0 introduced a new platform independent rendering engine, which works identical on Windows as well as on Mac OS X, Linux, Solaris and all other platforms. Due to the new rendering engine, text strings are computed a bit wider and a bit less in height than in v3.x. So in very rare cases it can happen that word breaks occur on different positions than in v3.x, which means that text objects might require more width than before, but also less height. For this reason we implemented a property, so you can switch back to the original rendering engine. This can be achieved by setting EngineRenderMode = VENGINE_RENDER_MODE_VER3. This is useful to adapt your existing code to this new version of VPE. But the old rendering engine is only available on Windows. **Therefore, for platform independence, it is strongly recommended to use the new rendering engine.** It is also likely that the old rendering engine will be removed in a future version of VPE (in several years).

**property EngineRenderMode [integer] VPE. EngineRenderMode**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VENGINE_RENDER_MODE_VER3</td>
<td>0</td>
<td>Ver3</td>
<td>VPE Renders Text compatible to Version 3.xx</td>
</tr>
<tr>
<td>VENGINE_RENDER_MODE_VER4</td>
<td>1</td>
<td>Ver4</td>
<td>VPE Renders Text compatible to Version 4.00 and higher</td>
</tr>
</tbody>
</table>

**Remarks:**

Reading an old v3.xx VPE document will turn VPE into VER3 Mode for the whole document!
9.4 PageBreak

Adds a new blank page to the end of the document. Then VPE sets internally the `CurrentPage` to this page, so that all further output-calls will draw onto this new page.

**method boolean VPE.PageBreak( )**

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success</td>
</tr>
<tr>
<td>False</td>
<td>failure</td>
</tr>
</tbody>
</table>

**Remarks:**

In case of an error, `LastError` is set. An error can occur, if the document is SwapFile based - i.e. you have set the property `SwapFileName` and there was an error writing to the SwapFile, because for example the harddisk is full.
9.5 AutoBreakMode

Controls, if and how text is handled, that overflows the bottom of the current output rectangle.

**property AutoBreakMode [integer] VPE.AutoBreakMode**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTO_BREAK_ON</td>
<td>On</td>
<td>Auto Break is activated. An Auto Break will happen if ( y_2 = VFREE ) or ( y &gt; VBOTTOMMARGIN ). Remaining text is broken onto the next page(s) with the following coordinates: ( x = ) the original ( x ) coordinate of the inserted object ( x_2 = ) the original ( x_2 ) coordinate of the inserted object ( y = ) top of the Output Rectangle of the successive page - if a new page is generated, it will be the top of the Default Output Rectangle ( y_2 = VFREE )</td>
</tr>
<tr>
<td>AUTO_BREAK_OFF</td>
<td>Off</td>
<td>Same behavior as AUTO_BREAK_ON (limited positioning / rendering to the bottom of the output rectangle is active), but remaining text is NOT broken onto next page(s). It is cut instead.</td>
</tr>
<tr>
<td>AUTO_BREAK_NO_LIMITS</td>
<td>NoLimits</td>
<td>Remaining text is NOT broken onto the next page(s), it can be placed anywhere on the paper with no limits.</td>
</tr>
<tr>
<td>AUTO_BREAK_FULL</td>
<td>Full</td>
<td>Auto Break is activated. An Auto Break will happen if ( y_2 = VFREE ) or ( y &gt; VBOTTOMMARGIN ). Remaining text is broken onto the next page(s) with the following coordinates: ( x = ) left margin of the Output Rectangle ( x_2 = ) right margin of the Output Rectangle ( y = ) top margin of the Output Rectangle ( y_2 = VFREE ) Note: if a new blank page is added by VPE, the Default Output Rectangle will be used to set ( x ), ( x_2 ) and ( y ). Otherwise if the next page is already existing, the Output Rectangle of the existing page is used. You can modify the Output Rectangle of an existing page at any time.</td>
</tr>
</tbody>
</table>

**Default:**

AUTO_BREAK_ON
9.6 PageCount
Retrieve the numbers of pages of the document

property long VPE.PageCount
read; runtime only

Returns:
The numbers of pages in the document
9.7 CurrentPage

Moves to the current active page, or returns the number of the current active page. This is the page where you can insert objects, all further output-calls will insert objects onto this page. It is independently from the Visual Page shown in the preview. (see "Multipage Documents" in the Programmer's Manual)

You can move at any time to any page to insert objects.

**property long VPE.CurrentPage**

  read / write; runtime only

**Possible Values:**

  The number of the current active page.

**Remarks:**

  In case of an error, LastError is set.

  If you want to show the preview and let the user work with it (e.g. scroll and print), while your application is still generating the document, see "Generating a Document while the Preview is open" in the Programmer's Manual for details.

**Examples:**

```
Doc.CurrentPage = 5    // moves to page 5
n = Doc.CurrentPage    // returns the value 5

Doc.CurrentPage = 5    // moves to page 5
Doc.Print(1, 1, "This is page 5")
Doc.CurrentPage = 2    // moves to page 2
Doc.Print(1, 1, "This is page 2")
```
9.8 PageFormat

Sets / returns the page format for the current page. You can set the page format for each page separately. All newly with PageBreak() added pages will have the page format which was specified at last.

property integer VPE.PageFormat

read / write; design & runtime

Possible Values:

one of the constants below

Default:

VPAPER_A4

Example:

VPE instructs the printer during printing, to switch the paper dimensions automatically according to the page dimensions of the VPE Document's currently printed page. Many printers can react on this and choose automatically the right paper from a different paper input bin. If you want to stop VPE from this automatism, you can specify the flag PRINT_NO_AUTO_PAGE_DIMS for the property PrintOptions.

Since the PageFormat and the Orientation of the document are independent from the settings of the printer, you can copy the settings of the printer to the document by using one of the following code samples:

Copying the printer's page format to the document from the default printer:

```vpe
VPE.OpenDoc  // the default printer is chosen automatically, so just copy the dimensions
VPE.PageWidth = VPE.DevPaperWidth
VPE.PageHeight = VPE.DevPaperHeight
VPE.PageOrientation = VPE.DevOrientation
```

Copying the printer's page format to the document from a specific printer:

```vpe
VPE.OpenDoc  // choose "Office Printer 2"
VPE.Device = "Office Printer 2"
VPE.PageWidth = VPE.DevPaperWidth
VPE.PageHeight = VPE.DevPaperHeight
VPE.PageOrientation = VPE.DevOrientation
```
Copying the printer's page format to the document from a PRS file:

**ActiveX / VCL:**

```cpp
VPE.OpenDoc
// use PRINT_NO_AUTO_PAGE_DIMS, in order to be able to retrieve the values
// for the page dimensions from the PRS file:
VPE.PrintOptions = PRINT_ALL + PRINT_NO_AUTO_PAGE_DIMS
VPE.SetupPrinter "personal settings.prs", PRINTDLG_ONFAIL
VPE.PageWidth = VPE.DevPaperWidth
VPE.PageHeight = VPE.DevPaperHeight
VPE.PageOrientation = VPE.DevOrientation

// switch PrintOptions back and let VPE control the printer's page
// dimensions, i.e. clear the flag PRINT_NO_AUTO_PAGE_DIMS
VPE.PrintOptions = PRINT_ALL
```

**.NET:**

```csharp
VPE.OpenDoc
// use PRINT_NO_AUTO_PAGE_DIMS, in order to be able to retrieve the values
// for the page dimensions from the PRS file:
VB: VPE.PrintOptions = PrintOptions.All + PrintOptions.NoAutoPageDims
C#: VPE.PrintOptions = PrintOptions.All | PrintOptions.NoAutoPageDims
VPE.SetupPrinter "personal settings.prs", PrintDialogControl.OnFail
VPE.PageWidth = VPE.DevPaperWidth
VPE.PageHeight = VPE.DevPaperHeight
VPE.PageOrientation = VPE.DevOrientation

// switch PrintOptions back and let VPE control the printer's page
// dimensions, i.e. clear the flag PRINT_NO_AUTO_PAGE_DIMS
VPE.PrintOptions = PrintOptions.All
```

You will notice that we are using `PageWidth` and `PageHeight` in the examples above. The reason is, that this circumvents a bug in many printer drivers, which return wrong values for `DevPaperFormat` in case a user defined format has been assigned to the printer. Because of this, the construction "VPE.PageFormat = VPE.DevPaperFormat" can not be used reliably. Instead, use the code from the examples above.

**ActiveX / VCL: PageFormat can be one of the following values:**

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPAPER_USER_DEFINED</td>
<td>0</td>
<td>User-Defined</td>
</tr>
<tr>
<td>VPAPER_A4</td>
<td>-1</td>
<td>A4 Sheet, 210- by 297-millimeters</td>
</tr>
<tr>
<td>VPAPER_LETTER</td>
<td>-2</td>
<td>US Letter, 8 1/2- by 11-inches</td>
</tr>
<tr>
<td>VPAPER_LEGAL</td>
<td>-3</td>
<td>Legal, 8 1/2- by 14-inches</td>
</tr>
<tr>
<td>VPAPER_CSHET</td>
<td>-4</td>
<td>C Sheet, 17- by 22-inches</td>
</tr>
<tr>
<td>VPAPER_DSHET</td>
<td>-5</td>
<td>D Sheet, 22- by 34-inches</td>
</tr>
<tr>
<td>VPAPER_ESHEET</td>
<td>-6</td>
<td>E Sheet, 34- by 44-inches</td>
</tr>
<tr>
<td>VPAPER_LETTERSMALL</td>
<td>-7</td>
<td>Letter Small, 8 1/2- by 11-inches</td>
</tr>
<tr>
<td>VPAPER_TABLOID</td>
<td>-8</td>
<td>Tabloid, 11- by 17-inches</td>
</tr>
<tr>
<td>VPAPER_LEDGER</td>
<td>-9</td>
<td>Ledger, 17- by 11-inches</td>
</tr>
<tr>
<td>VPAPER_STATEMENT</td>
<td>-10</td>
<td>Statement, 5 1/2- by 8 1/2-inches</td>
</tr>
<tr>
<td>VPAPER_EXECUTIVE</td>
<td>-11</td>
<td>Executive, 7 1/4- by 10 1/2-inches</td>
</tr>
<tr>
<td>VPAPER_A3</td>
<td>-12</td>
<td>A3 sheet, 297- by 420-millimeters</td>
</tr>
<tr>
<td>VPAPER_A4SMALL</td>
<td>-13</td>
<td>A4 small sheet, 210- by 297-millimeters</td>
</tr>
<tr>
<td>VPAPER_A5</td>
<td>-14</td>
<td>A5 sheet, 148- by 210-millimeters</td>
</tr>
<tr>
<td>VPAPER_B4</td>
<td>-15</td>
<td>B4 sheet, 250- by 354-millimeters</td>
</tr>
<tr>
<td>VPAPER_B5</td>
<td>-16</td>
<td>B5 sheet, 182- by 257-millimeter paper</td>
</tr>
<tr>
<td>-------------------</td>
<td>------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>VPAPER_FOLIO</td>
<td>-17</td>
<td>Folio, 8 1/2- by 13-inch paper</td>
</tr>
<tr>
<td>VPAPER_QUARTO</td>
<td>-18</td>
<td>Quarto, 215- by 275-millimeter paper</td>
</tr>
<tr>
<td>VPAPER_10X14</td>
<td>-19</td>
<td>10- by 14-inch sheet</td>
</tr>
<tr>
<td>VPAPER_11X17</td>
<td>-20</td>
<td>11- by 17-inch sheet</td>
</tr>
<tr>
<td>VPAPER_NOTE</td>
<td>-21</td>
<td>Note, 8 1/2- by 11-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_9</td>
<td>-22</td>
<td>#9 Envelope, 3 7/8- by 8 7/8-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_10</td>
<td>-23</td>
<td>#10 Envelope, 4 1/8- by 9 1/2-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_11</td>
<td>-24</td>
<td>#11 Envelope, 4 1/2- by 10 3/8-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_12</td>
<td>-25</td>
<td>#12 Envelope, 4 3/4- by 11-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_14</td>
<td>-26</td>
<td>#14 Envelope, 5- by 11 1/2-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_DL</td>
<td>-27</td>
<td>DL Envelope, 110- by 220-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_C5</td>
<td>-28</td>
<td>C5 Envelope, 162- by 229-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_C3</td>
<td>-29</td>
<td>C3 Envelope, 324- by 458-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_C4</td>
<td>-30</td>
<td>C4 Envelope, 229- by 324-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_C6</td>
<td>-31</td>
<td>C6 Envelope, 114- by 162-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_C65</td>
<td>-32</td>
<td>C65 Envelope, 114- by 229-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_B4</td>
<td>-33</td>
<td>B4 Envelope, 250- by 353-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_B5</td>
<td>-34</td>
<td>B5 Envelope, 176- by 250-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_B6</td>
<td>-35</td>
<td>B6 Envelope, 176- by 125-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_ITALY</td>
<td>-36</td>
<td>Italy Envelope, 110- by 230-millimeters</td>
</tr>
<tr>
<td>VPAPER_ENV_MONARCH</td>
<td>-37</td>
<td>Monarch Envelope, 3 7/8- by 7 ½-inches</td>
</tr>
<tr>
<td>VPAPER_ENV_PERSONAL</td>
<td>-38</td>
<td>6 3/4 Envelope, 3 5/8- by 6 1/2-inches</td>
</tr>
<tr>
<td>VPAPER_FANFOLD_US</td>
<td>-39</td>
<td>US Std Fanfold, 14 7/8- by 11-inches</td>
</tr>
<tr>
<td>VPAPER_FANFOLD_STD_GERMAN</td>
<td>-40</td>
<td>German Std Fanfold, 8 1/2- by 12-inches</td>
</tr>
<tr>
<td>VPAPER_FANFOLD_LGL_GERMAN</td>
<td>-41</td>
<td>German Legal Fanfold, 8 1/2- by 13-inches</td>
</tr>
<tr>
<td>VPAPER_ISO_B4</td>
<td>-42</td>
<td>B4 (ISO) 250 x 353 mm</td>
</tr>
<tr>
<td>VPAPER_JAPANESE_POSTCARD</td>
<td>-43</td>
<td>Japanese Postcard 100 x 148 mm</td>
</tr>
<tr>
<td>VPAPER_9X11</td>
<td>-44</td>
<td>9 x 11 in</td>
</tr>
<tr>
<td>VPAPER_10X11</td>
<td>-45</td>
<td>10 x 11 in</td>
</tr>
<tr>
<td>VPAPER_15X11</td>
<td>-46</td>
<td>15 x 11 in</td>
</tr>
<tr>
<td>VPAPER_ENV_INVITE</td>
<td>-47</td>
<td>Envelope Invite 220 x 220 mm</td>
</tr>
<tr>
<td>VPAPER_RESERVED_48</td>
<td>-48</td>
<td>RESERVED--DO NOT USE</td>
</tr>
<tr>
<td>VPAPER_RESERVED_49</td>
<td>-49</td>
<td>RESERVED--DO NOT USE</td>
</tr>
<tr>
<td>VPAPER_LETTER_EXTRA</td>
<td>-50</td>
<td>Letter Extra 9 (\frac{275}{x}) x 12 in</td>
</tr>
<tr>
<td>VPAPER_LEGAL_EXTRA</td>
<td>-51</td>
<td>Legal Extra 9 (\frac{275}{x}) x 15 in</td>
</tr>
<tr>
<td>VPAPER_TABLOYD_EXTRA</td>
<td>-52</td>
<td>Tabloid Extra 11.69 x 18 in</td>
</tr>
<tr>
<td>VPAPER_A4_EXTRA</td>
<td>-53</td>
<td>A4 Extra 9.27 x 12.69 in</td>
</tr>
<tr>
<td>VPAPER_LETTER_TRANSVERSE</td>
<td>-54</td>
<td>Letter Transverse 8 (\frac{275}{x}) x 11 in</td>
</tr>
<tr>
<td>VPAPER_A4_TRANSVERSE</td>
<td>-55</td>
<td>A4 Transverse 210 x 297 mm</td>
</tr>
<tr>
<td>VPAPER_LETTER_EXTRA_TRANSVERSE</td>
<td>-56</td>
<td>Letter Extra Transverse 9(275) x 12 in</td>
</tr>
<tr>
<td>VPAPER_A_PLUS</td>
<td>-57</td>
<td>SuperA/ SuperA/ A4 227 x 356 mm</td>
</tr>
<tr>
<td>VPAPER_B_PLUS</td>
<td>-58</td>
<td>SuperB/ SuperB/ A3 305 x 487 mm</td>
</tr>
<tr>
<td>VPAPER_LETTER_PLUS</td>
<td>-59</td>
<td>Letter Plus 8.5 x 12.69 in</td>
</tr>
<tr>
<td>VPAPER_A4_PLUS</td>
<td>-60</td>
<td>A4 Plus 210 x 330 mm</td>
</tr>
<tr>
<td>VPAPER_A5_TRANSVERSE</td>
<td>-61</td>
<td>A5 Transverse 148 x 210 mm</td>
</tr>
<tr>
<td>VPAPER_B5_TRANSVERSE</td>
<td>-62</td>
<td>B5 (JIS) Transverse 182 x 257 mm</td>
</tr>
<tr>
<td>VPAPER_A3_EXTRA</td>
<td>-63</td>
<td>A3 Extra 322 x 445 mm</td>
</tr>
<tr>
<td>Layout Function</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>--------------------------</td>
<td></td>
</tr>
<tr>
<td>VPAPER_A5_EXTRA</td>
<td>64 A5 Extra 174 x 235 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_B5_EXTRA</td>
<td>65 B5 (ISO) Extra 201 x 276 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_A2</td>
<td>66 A2 420 x 594 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_A3_TRANSVERSE</td>
<td>67 A3 Transverse 297 x 420 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_A3_EXTRA_TRANSVERSE</td>
<td>68 A3 Extra Transverse 322 x 445 mm</td>
<td></td>
</tr>
<tr>
<td>Windows 2000 or Higher:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPAPER_DBL_JAPANESE_POSTCARD</td>
<td>-69 Japanese Double Postcard 200 x 148 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_A6</td>
<td>-70 A6 105 x 148 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_JENV_KAKU2</td>
<td>-71 Japanese Envelope Kaku #2</td>
<td></td>
</tr>
<tr>
<td>VPAPER_JENV_KAKU3</td>
<td>-72 Japanese Envelope Kaku #3</td>
<td></td>
</tr>
<tr>
<td>VPAPER_JENV_CHOU3</td>
<td>-73 Japanese Envelope Chou #3</td>
<td></td>
</tr>
<tr>
<td>VPAPER_JENV_CHOU4</td>
<td>-74 Japanese Envelope Chou #4</td>
<td></td>
</tr>
<tr>
<td>VPAPER_LETTER_ROTATED</td>
<td>-75 Letter Rotated 11 x 8 1/2 in</td>
<td></td>
</tr>
<tr>
<td>VPAPER_A3_ROTATED</td>
<td>-76 A3 Rotated 420 x 297 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_A4_ROTATED</td>
<td>-77 A4 Rotated 297 x 210 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_A5_ROTATED</td>
<td>-78 A5 Rotated 210 x 148 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_B4_JIS_ROTATED</td>
<td>-79 B4 (JIS) Rotated 364 x 257 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_B5_JIS_ROTATED</td>
<td>-80 B5 (JIS) Rotated 257 x 182 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_JAPANESE_POSTCARD_ROTATED</td>
<td>-81 Japanese Postcard Rotated 148 x 100 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_DBL_JAPANESE_POSTCARD_ROTATED</td>
<td>-82 Double Japanese Postcard Rotated 148 x 200mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_A6_ROTATED</td>
<td>-83 A6 Rotated 148 x 105 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_JENV_KAKU2_ROTATED</td>
<td>-84 Japanese Envelope Kaku #2 Rotated</td>
<td></td>
</tr>
<tr>
<td>VPAPER_JENV_KAKU3_ROTATED -85</td>
<td>-85 Japanese Envelope Kaku #3 Rotated</td>
<td></td>
</tr>
<tr>
<td>VPAPER_JENV_CHOU3_ROTATED</td>
<td>-86 Japanese Envelope Chou #3 Rotated</td>
<td></td>
</tr>
<tr>
<td>VPAPER_JENV_CHOU4_ROTATED</td>
<td>-87 Japanese Envelope Chou #4 Rotated</td>
<td></td>
</tr>
<tr>
<td>VPAPER_B6_JIS</td>
<td>-88 B6 (JIS) 128 x 182 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_B6_JIS_ROTATED</td>
<td>-89 B6 (JIS) Rotated 182 x 128 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_12X11</td>
<td>-90 12 x 11 in</td>
<td></td>
</tr>
<tr>
<td>VPAPER_JENV_YOU4</td>
<td>-91 Japanese Envelope You #4</td>
<td></td>
</tr>
<tr>
<td>VPAPER_JENV_YOU4_ROTATED</td>
<td>-92 Japanese Envelope You #4 Rotated</td>
<td></td>
</tr>
<tr>
<td>VPAPER_P16K</td>
<td>-93 PRC 16K 146 x 215 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_P32K</td>
<td>-94 PRC 32K 97 x 151 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_P32KBIG</td>
<td>-95 PRC 32K(Big) 97 x 151 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_1</td>
<td>-96 PRC Envelope #1 102 x 165 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_2</td>
<td>-97 PRC Envelope #2 102 x 176 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_3</td>
<td>-98 PRC Envelope #3 125 x 176 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_4</td>
<td>-99 PRC Envelope #4 110 x 208 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_5</td>
<td>-100 PRC Envelope #5 110 x 220 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_6</td>
<td>-101 PRC Envelope #6 120 x 230 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_7</td>
<td>-102 PRC Envelope #7 160 x 230 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_8</td>
<td>-103 PRC Envelope #8 120 x 309 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_9</td>
<td>-104 PRC Envelope #9 229 x 324 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_10</td>
<td>-105 PRC Envelope #10 324 x 458 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_P16K_ROTATED</td>
<td>-106 PRC 16K Rotated</td>
<td></td>
</tr>
<tr>
<td>VPAPER_P32K_ROTATED</td>
<td>-107 PRC 32K Rotated</td>
<td></td>
</tr>
<tr>
<td>VPAPER_P32KBIG_ROTATED</td>
<td>-108 PRC 32K(Big) Rotated</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_1_ROTATED</td>
<td>-109 PRC Envelope #1 Rotated 165 x 102 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_2_ROTATED</td>
<td>-110 PRC Envelope #2 Rotated 176 x 102 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_3_ROTATED</td>
<td>PRC Envelope #3 Rotated 176 x 125 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_4_ROTATED</td>
<td>PRC Envelope #4 Rotated 208 x 110 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_5_ROTATED</td>
<td>PRC Envelope #5 Rotated 220 x 110 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_6_ROTATED</td>
<td>PRC Envelope #6 Rotated 230 x 120 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_7_ROTATED</td>
<td>PRC Envelope #7 Rotated 230 x 160 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_8_ROTATED</td>
<td>PRC Envelope #8 Rotated 309 x 120 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_9_ROTATED</td>
<td>PRC Envelope #9 Rotated 324 x 229 mm</td>
<td></td>
</tr>
<tr>
<td>VPAPER_PENV_10_ROTATED</td>
<td>PRC Envelope #10 Rotated 458 x 324 mm</td>
<td></td>
</tr>
</tbody>
</table>

`.NET, Java, PHP, Python, Ruby, etc.: PageFormat can be one of the following values

public enum PageFormat
{
    A4, // A4 Sheet, 210- by 297-millimeters
    Letter, // US Letter, 8 1/2- by 11-inches
    Legal, // Legal, 8 1/2- by 14-inches
    CSheet, // C Sheet, 17- by 22-inches
    DSheet, // D Sheet, 22- by 34-inches
    ESheet, // E Sheet, 34- by 44-inches
    LetterSmall, // Letter Small, 8 1/2- by 11-inches
    Tabloid, // Tabloid, 11- by 17-inches
    Ledger, // Ledger, 17- by 11-inches
    Statement, // Statement, 5 1/2- by 8 1/2-inches
    Executive, // Executive, 7 1/4- by 10 1/2-inches
    A3, // A3 sheet, 297- by 420-millimeters
    A4Small, // A4 small sheet, 210- by 297-millimeters
    A5, // A5 sheet, 148- by 210-millimeters
    B4, // B4 sheet, 250- by 354-millimeters
    B5, // B5 sheet, 182- by 257-millimeter paper
    Folio, // Folio, 8 1/2- by 13-inch paper
    Quarto, // Quarto, 215- by 275-millimeter paper
    Standard10x14, // 10- by 14-inch sheet
    Standard11x17, // 11- by 17-inch sheet
    Note, // Note, 8 1/2- by 11-inches
    Envelope9, // #9 Envelope, 3 7/8- by 8 7/8-inches
    Envelope10, // #10 Envelope, 4 1/8- by 9 1/2-inches
    Envelope11, // #11 Envelope, 4 1/2- by 10 3/8-inches
    Envelope12, // #12 Envelope, 4 3/4- by 11-inches
    Envelope14, // #14 Envelope, 5- by 11 1/2-inches
    EnvelopeDL, // DL Envelope, 110- by 220-millimeters
    EnvelopeC5, // C5 Envelope, 162- by 229-millimeters
    EnvelopeC3, // C3 Envelope, 324- by 458-millimeters
    EnvelopeC4, // C4 Envelope, 229- by 324-millimeters
    EnvelopeC6, // C6 Envelope, 114- by 162-millimeters
    EnvelopeC65, // C65 Envelope, 114- by 229-millimeters
    EnvelopeB4, // B4 Envelope, 250- by 353-millimeters
    EnvelopeB5, // B5 Envelope, 176- by 250-millimeters
    EnvelopeB6, // B6 Envelope, 176- by 125-millimeters
    EnvelopeItaly, // Italy Envelope, 110- by 230-millimeters
    EnvelopeMonarch, // Monarch Envelope, 3 7/8- by 7 1/2-inches
}
EnvelopePersonal, 6 3/4 Envelope, 3 5/8- by 6 1/2-inches
FanfoldUS, US Std Fanfold, 14 7/8- by 11-inches
FanfoldStdGerman, German Std Fanfold, 8 1/2- by 12-inches
FanfoldLglGerman, German Legal Fanfold, 8 1/2- by 13-inches
UserDefined, User Defined
IsoB4, B4 (ISO) 250 x 353 mm
JapanesePostcard, Japanese Postcard 100 x 148 mm
Standard9x11, 9 x 11 in
Standard10x11, 10 x 11 in
Standard15x11, 15 x 11 in
EnvelopeInvite, Envelope Invite 220 x 220 mm
Reserved48, RESERVED--DO NOT USE
Reserved49, RESERVED--DO NOT USE
LetterExtra, Letter Extra 9 275 x 12 in
LegalExtra, Legal Extra 9 275 x 15 in
TabloidExtra, Tabloid Extra 11.69 x 18 in
A4Extra, A4 Extra 9.27 x 12.69 in
LetterTransverse, Letter Transverse 8 275 x 11 in
A4Transverse, A4 Transverse 210 x 297 mm
LetterExtraTransverse, Letter Extra Transverse 9 275 x 12 in
APlus, SuperA/SuperA/A4 227 x 356 mm
BPlus, SuperB/SuperB/A3 305 x 487 mm
LetterPlus, Letter Plus 8.5 x 12.69 in
A4Plus, A4 Plus 210 x 330 mm
A5Transverse, A5 Transverse 148 x 210 mm
B5Transverse, B5 (JIS) Transverse 182 x 257 mm
A3Extra, A3 Extra 322 x 445 mm
A5Extra, A5 Extra 174 x 235 mm
B5Extra, B5 (ISO) Extra 201 x 276 mm
A2, A2 420 x 594 mm
A3Transverse, A3 Transverse 297 x 420 mm
A3ExtraTransverse, A3 Extra Transverse 322 x 445 mm

Windows 2000 or Higher:
DbJapanesePostcard, Japanese Double Postcard 200 x 148 mm
A6, A6 105 x 148 mm
JapaneseEnvelopeKaku2, Japanese Envelope Kaku #2
JapaneseEnvelopeKaku3, Japanese Envelope Kaku #3
JapaneseEnvelopeChou3, Japanese Envelope Chou #3
JapaneseEnvelopeChou4, Japanese Envelope Chou #4
LetterRotated, Letter Rotated 11 x 8 1/2 in
A3Rotated, A3 Rotated 420 x 297 mm
A4Rotated, A4 Rotated 297 x 210 mm
A5Rotated, A5 Rotated 210 x 148 mm
B4JisRotated, B4 (JIS) Rotated 364 x 257 mm
B5JisRotated, B5 (JIS) Rotated 257 x 182 mm
JapanesePostcardRotated, Japanese Postcard Rotated 148 x 100 mm
DbJapanesePostcardRotated, Double Japanese Postcard Rotated 148 x 200 mm
A6Rotated, A6 Rotated 148 x 105 mm
JapaneseEnvelopeKaku2Rotated, Japanese Envelope Kaku #2 Rotated
JapaneseEnvelopeKaku3Rotated, Japanese Envelope Kaku #3 Rotated
JapaneseEnvelopeChou3Rotated, Japanese Envelope Chou #3 Rotated
JapaneseEnvelopeChou4Rotated, Japanese Envelope Chou #4 Rotated
B6Jis, B6 (JIS) 128 x 182 mm
B6JisRotated, B6 (JIS) Rotated 182 x 128 mm
Standard12x11, 12 x 11 in
JapaneseEnvelopeYou4, Japanese Envelope You #4
JapaneseEnvelopeYou4Rotated, Japanese Envelope You #4 Rotated
Prc16K, PRC 16K 146 x 215 mm
Prc32K, PRC 32K 97 x 151 mm
Prc32KBig, PRC 32K(Big) 97 x 151 mm
PrcEnvelope1, PRC Envelope #1 102 x 165 mm
PrcEnvelope2, PRC Envelope #2 102 x 176 mm
PrcEnvelope3, PRC Envelope #3 125 x 176 mm
PrcEnvelope4, PRC Envelope #4 110 x 208 mm
PrcEnvelope5, PRC Envelope #5 110 x 220 mm
PrcEnvelope6, PRC Envelope #6 120 x 230 mm
PrcEnvelope7, PRC Envelope #7 160 x 230 mm
PrcEnvelope8, PRC Envelope #8 120 x 309 mm
PrcEnvelope9, PRC Envelope #9 229 x 324 mm
PrcEnvelope10, PRC Envelope #10 324 x 458 mm
Prc16KRotated, PRC 16K Rotated
Prc32KRotated, PRC 32K Rotated
Prc32KBigRotated, PRC 32K(Big) Rotated
PrcEnvelope1Rotated, PRC Envelope #1 Rotated 165 x 102 mm
PrcEnvelope2Rotated, PRC Envelope #2 Rotated 176 x 102 mm
PrcEnvelope3Rotated, PRC Envelope #3 Rotated 176 x 125 mm
PrcEnvelope4Rotated, PRC Envelope #4 Rotated 208 x 110 mm
PrcEnvelope5Rotated, PRC Envelope #5 Rotated 220 x 110 mm
PrcEnvelope6Rotated, PRC Envelope #6 Rotated 230 x 120 mm
PrcEnvelope7Rotated, PRC Envelope #7 Rotated 230 x 160 mm
PrcEnvelope8Rotated, PRC Envelope #8 Rotated 309 x 120 mm
PrcEnvelope9Rotated, PRC Envelope #9 Rotated 324 x 229 mm
PrcEnvelope10Rotated, PRC Envelope #10 Rotated 458 x 324 mm
9.9 PageWidth

Sets / returns the page width for the current page to a user defined value. Also all newly with PageBreak() added pages will have this width.

property VpeCoord VPE.PageWidth

read / write; design & runtime

Possible Values:
the width of the current page

Default:
21 (21cm = width of VPAPER_A4)

Remarks:
see PageFormat

VPE instructs the printer during printing, to switch the paper dimensions automatically according to the page dimensions of the VPE Document's currently printed page. Many printers can react on this and choose automatically the right paper from a different paper input bin. If you want to stop VPE from this automatism, you can specify the flag PRINT_NO_AUTO_PAGE_DIMS for the property PrintOptions.

In order to let the printer accept the automatic selection of user defined page formats (i.e. you are not specifying PRINT_NO_AUTO_PAGE_DIMS), it might be necessary to define a form of the desired page format.

Example: with "Start Menu | Settings | Printers" the window with the installed printers will appear. Right-click on a blank area of the window and choose "Server Properties" from the pop-up menu. In the upcoming dialog, define a custom form of the desired dimensions. For example name it "Test" and set the width to 760 and height to 1280.

In your source code, select the printer and the desired page format like this:

```vpe
Vpe.Device = "Epson LQ-550";
Vpe.PageWidth = 7.60;
Vpe.PageHeight = 12.80;
```

That's it.

We experienced that this property doesn’t work with some printer drivers. We tested this for example on an Epson LQ-550 dot-matrix printer and it didn’t work on WfW 3.11 and NT 3.51, but it worked on Win95. But we heared about, that a LQ-510 printer driver will solve the problem for the LQ-550. So if your printer does not respond to this setting, it is a printer driver problem. In that case you can try to use another compatible printer driver.

The page size is freely definable up to 999cm x 999cm.

Win 9x / Me: on Win 9x/Me the GDI has 16-bit coordinates, so the dimensions of a page may not exceed 32.7 cm (12.9 inch) when using the version 4 renderer, and 138.7 cm (54.6 inch) when using the version 3 renderer. Those values apply to the preview. For other output devices, e.g. printers, the dimensions of a page may not exceed (32767 / DPI) * 2.54 cm, where DPI = the resolution of the output device.

Example: for a printer with 1200 DPI resolution, the maximum page dimensions may be (32767 / 1200) * 2.54 cm = 69.35 cm.
9.10 PageHeight

Sets the page height for the current page to a user defined value. Also all newly with PageBreak() added pages will have this page format.

**property VpeCoord VPE.PageHeight**

  read / write; design & runtime

**Possible Values:**

  the height of the current page

**Default:**

  29.70 (29.7cm = height of VPAPER_A4)

**Remarks:**

  see PageFormat
  see PageWidth
9.11 PageOrientation

Sets / returns the page orientation for the current page. Also all newly with PageBreak() added pages will have this orientation. The orientation is reflected in the preview if PaperView = True. Also the printer will print the page in the specified orientation.

property PageOrientation [integer] VPE.PageOrientation

read / write; design & runtime

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VORIENT_PORTRAIT</td>
<td>1</td>
<td>Portrait</td>
<td></td>
</tr>
<tr>
<td>VORIENT_LANDSCAPE</td>
<td>2</td>
<td>Landscape</td>
<td></td>
</tr>
</tbody>
</table>

Remarks:
Changing the orientation also modifies the OutRect of the current page and the DefOutRect (see also: "Page Margins" in the Programmer's Manual). The rectangles are rotated in a way that the margin spaces are kept the same. For example, if the OutRect and DefOutRect are set in a way, that all margins are 2cm from the paper borders, changing the orientation will keep those borders by rotating the rectangles.

This property is independent from DevOrientation. You should always use this function to specify the orientation.

Example:

**ActiveX / VCL:**
```
VPE.OpenDoc
VPE.SetupPrinter("personal settings.prs", PRINTDLG_ONFAIL)
VPE.PageFormat = VPE.DevPaperFormat
VPE.PageOrientation = VPE.DevOrientation
```

**.NET:**
```
VPE.OpenDoc
VPE.SetupPrinter("personal settings.prs", PrintDialogControl.OnFail)
VPE.PageFormat = VPE.DevPaperFormat
VPE.PageOrientation = VPE.DevOrientation
```

Changing the orientation during the print-job doesn’t work with some (buggy) printer drivers, for example some fax drivers and the HP4 M Plus driver (the HP4 PS driver should work with the HP4 M Plus printer!). Changing the orientation with such drivers on other pages than the very first page might not work. Most printer drivers will work.

*Printer drivers are manufactured by vendors independent of IDEAL Software; we make no warranty, implied or otherwise, regarding these product’s performance or reliability.*
Example:

**ActiveX / VCL:**

```plaintext
Doc.DevOrientation = VORIENT_LANDSCAPE
Doc.PageBreak()
Doc.DevOrientation = VORIENT_PORTrait
Doc.PageBreak()
Doc.DevOrientation = VORIENT_LANDSCAPE
```

**.NET:**

```plaintext
Doc.DevOrientation = PageOrientation.Landscape
Doc.PageBreak()
Doc.PageBreak()
Doc.DevOrientation = PageOrientation.Landscape
```

This example sets the orientation for the current page to Landscape, adds a new page and sets the Orientation of the new added page to Portrait and adds a third page and sets the Orientation of the new added page back to Landscape.
9.12 PaperBin

[Not supported by the Community Edition]

Sets the paper bin for the current page. Also all newly with `PageBreak()` added pages will print to this bin.

**property PaperBin [integer] VPE.PaperBin**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBIN_UNTOUCHED</td>
<td>0</td>
<td>Untouched</td>
<td></td>
</tr>
<tr>
<td>VBIN_UPPER</td>
<td>1</td>
<td>Upper</td>
<td></td>
</tr>
<tr>
<td>VBIN_ONLYONE</td>
<td>1</td>
<td>OnlyOne</td>
<td></td>
</tr>
<tr>
<td>VBIN_LOWER</td>
<td>2</td>
<td>Lower</td>
<td></td>
</tr>
<tr>
<td>VBIN_MIDDLE</td>
<td>3</td>
<td>Middle</td>
<td></td>
</tr>
<tr>
<td>VBIN_MANUAL</td>
<td>4</td>
<td>Manual</td>
<td></td>
</tr>
<tr>
<td>VBIN_ENVELOPE</td>
<td>5</td>
<td>Envelope</td>
<td></td>
</tr>
<tr>
<td>VBIN_ENVMANUAL</td>
<td>6</td>
<td>EnvelopeManual</td>
<td></td>
</tr>
<tr>
<td>VBIN_AUTO</td>
<td>7</td>
<td>Auto</td>
<td></td>
</tr>
<tr>
<td>VBIN_TRACTOR</td>
<td>8</td>
<td>Tractor</td>
<td></td>
</tr>
<tr>
<td>VBIN_SMALLFMT</td>
<td>9</td>
<td>SmallFormat</td>
<td></td>
</tr>
<tr>
<td>VBIN_LARGEFMT</td>
<td>10</td>
<td>LargeFormat</td>
<td></td>
</tr>
<tr>
<td>VBIN_LARGECAPACITY</td>
<td>11</td>
<td>LargeCapacity</td>
<td></td>
</tr>
<tr>
<td>VBIN_CASSETTE</td>
<td>14</td>
<td>Cassette</td>
<td></td>
</tr>
</tbody>
</table>

Not all of the bin options are available on every printer. Check the printer documentation for more specific descriptions of these options. In addition the PaperBin-ID constants above can not be used reliably. Instead enumerate the available paper bins for the selected printer and use `GetDevPaperBinID` (please see below for details).

**Default:**

VBIN_UNTOUCHED

**Remarks:**

This property is independent from `DevPaperBin`. You should not use DevPaperBin, instead always use this property to specify the paper bin.

The value VBIN_UNTOUCHED is a VPE internal constant. It instructs VPE not to change the bin from the setting the current selected device has.

**This property conflicts with the default behavior of VPE**, to provide the page dimensions of the currently printed page to the printer driver. Many printer drivers choose automatically the right paper from a different paper input bin. To override the default behavior of VPE and to make the PaperBin property work, you have to specify the flag `PRINT_NO_AUTO_PAGE_DIMS` for the property `PrintOptions`. 
Do not rely on the bin names of the constants, like VBIN_UPPER, etc. - their names might not match correctly a tray: for example under WinNT for an HP 5P you can select the lower tray with VBIN_LOWER and the upper tray with VBIN_UPPER, but under Win95 for the same printer you can only use VBIN_MANUAL for the upper tray and VBIN_UPPER for the lower tray (yes, it is not a mistake: the LOWER tray is selected with VBIN_UPPER!). Solution: operate with BIN-ID's (see DevEnumPaperBins) and ff.

Changing the bin during the print-job doesn't work with some (buggy) printer drivers. Changing the bin with such drivers for other pages than the very first page might not work. Our own tests revealed for example that the HP 2200 D driver can only switch once the bin, but thereafter it will not switch the bin again. The HP 5P driver behaves correctly unless multiple copies and collation are selected. If collation is activated, the driver will print all pages except the first multiple times as if non-collation was selected.

Printer drivers are manufactured by vendors independent of IDEAL Software; we make no warranty, implied or otherwise, regarding these product's performance or reliability.

Example:

**ActiveX / VCL:**

```pascal
Doc.PaperBin = VBIN_UPPER
```

Will instruct the printer during printing, to print this page on the upper paper bin (if available).

```pascal
Doc.PageBreak()
Doc.PaperBin = VBIN_LOWER
```

Adds a new page and will instruct the printer during printing, to print this page on the lower paper bin (if available).

```pascal
Doc.PageBreak()
Doc.PaperBin = VBIN_UPPER
```

Adds a third page and will instruct the printer during printing, to print this page on the upper paper bin again (if available).

**.NET:**

```csharp
```

Will instruct the printer during printing, to print this page on the upper paper bin (if available).

```csharp
Doc.PageBreak()
Doc.PaperBin = PaperBin.Lower
```

Adds a new page and will instruct the printer during printing, to print this page on the lower paper bin (if available).

```csharp
Doc.PageBreak()
```

Adds a third page and will instruct the printer during printing, to print this page on the upper paper bin again (if available).
9.13 StoreSet

All current property settings (pen-size, alignment, colors, font, etc.) are stored in a buffer under the specified ID. You can create as much buffers as you like (only limited by available memory). To have access to the different buffers, you need to specify a unique ID for each. This is useful if you want to switch back to the current settings later again.

```java
method void VPE.StoreSet(
    long ID
)
```

*long ID*

the ID under which you store the properties

**Remarks:**
The following properties are stored:

- `FontName`
- `FontSize`
- `PenSize`
- `PenStyle`
- `PenColor`
- `Bold`
- `Italic`
- `Underline`
- `StrikeOut`
- `TextAlignment`
- `TextColor`
- `BkgColor`
- `BkgMode`
- `GradientStartColor`
- `GradientEndColor`
- `GradientTriColor`
- `GradientMiddleColorPosition`
- `GradientMiddleColor`
- `GradientRotation`
- `HatchStyle`
- `HatchColor`
- `CornerRadius`
- `AutoBreakMode`
- `Rotation`
- `PicturePage`
- `PictureType`
- `PictureKeepAspect`
- `PictureCache`
- `PictureScale2Gray`
- `PictureScale2GrayFloat`
- `PictureX2YResolution`
- `PictureBestFit`
- `PictureEmbedInDoc`
- `PictureDrawExact`
- `PictureDefaultDPIX`
- `PictureDefaultDPIY`
JpegExportOptions
TiffExportOptions
BmpExportOptions
PnmExportOptions
GifExportOptions
PictureExportColorDepth
PictureExportDither

Charset
CharPlacement
InsertAtBottomZOrder

RTFParagraph - The complete Paragraph Settings
ChartProperties

BarcodeMainText
BarcodeAddText
BarcodeAlignment
BarcodeAutoChecksum
BarcodeThinBar
BarcodeThickBar

Viewable
Printable
Streamable
Shadowed

CharacterCount
SubdividerPenSize
SubdividerPenColor
AltSubdividerNPosition
AltSubdividerPenSize
AltSubdividerPenColor
BottomLinePenSize
BottomLinePenColor
SubdividerStyle
AltSubdividerStyle
EditFlags

BookmarkDestination
BookmarkStyle
BookmarkColor

Bar2DAlignment
DataMatrixEncodingFormat
DataMatrixEccType
DataMatrixRows
DataMatrixColumns
DataMatrixMirror
DataMatrixBorder
QRCodeVersion
QRCodeEccLevel
QRCodeMode
QRCodeBorder
PDF417ErrorLevel
PDF417Columns
PDF417Rows
AztecFlags
AztecControl
AztecMenu
AztecMultipleSymbols
AztecID

Example:

```
Doc.StoreSet(1)     // store the current settings
Doc.FontSize = 12   // modify the current properties,
Doc.PenSize = 6     // and output some text
Doc.WriteBox(1, 1, "Hello World!")
Doc.UseSet(1)       // return to the original settings
Doc.RemoveSet(1)    // delete the stored settings
```
9.14 UseSet

This resets all properties to the stored values.

\[
\text{method void VPE.UseSet(}
\quad \text{long } ID
\quad \}
\]

\textit{long ID}
the ID under which you stored the properties

\textbf{See also:}
\begin{itemize}
\item StoreSet
\end{itemize}
9.15 RemoveSet

Removes the setting specified under ID from memory.

```java
method void VPE.RemoveSet(
    long ID
)

long ID
    the ID under which you stored the properties
```

**See also:**

- StoreSet
9.16 nLeft, nTop, nRight, nBottom

Returns / sets the specific coordinate of the last inserted object. Setting these properties is also possible. You may also use the V-Flags VLEFT, VTOP, VRIGHT, VBOTTOM, if you don't want to compute offsets. For the .NET component we always recommend to use the n-Properties instead of the V-Flags.


property VpeCoord VPE.nLeft
property VpeCoord VPE.nTop
property VpeCoord VPE.nRight
property VpeCoord VPE.nBottom

    read / write; runtime only

Possible Values:
The coordinates of the last inserted object.

Example:

```
Doc.Write(1, 2, 4, 3, "Hello")
```

After executing this method the properties contain the following values:
nLeft = 1, nTop = 2, nRight = 4, nBottom = 3

You can insert an object at the right border of the last inserted object with:

```
Doc.Print(Doc.nRight, Doc.nTop, " World!")
```

The equal result (but with faster processing) is created with the following statement:

ActiveX / VCL:

```
Doc.Print(VRIGHT, VTOP, " World!")
```

.NET:

```
Doc.Print(VpeControl.VRIGHT, VpeControl.VTOP, " World!")
```

Note: for the .NET component we always recommend to use the n-Properties instead of the V-Flags.

If you want to insert the second object with an offset, you can only use:

```
Doc.Print(Doc.nRight + 0.5, Doc.nTop, " World!")
```

which will insert the object with an offset of 0.5 cm to the right border of the previously inserted object.
The following is not possible:

```vbnet
Doc.Print(VRIGHT + 0.5, VTOP, " World!")
```

because VRIGHT is a constant (its value is -2147483551) which instructs VPE to retrieve the value of the property nRight internally. If you add 0.5, the result is -2147483550.5, which is interpreted as a standard coordinate.

Remarks:
For completeness, we list the values of the V-Flags here.
You should always use the named constants instead of the values.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLEFT</td>
<td>-2147483550</td>
</tr>
<tr>
<td>VRIGHT</td>
<td>-2147483551</td>
</tr>
<tr>
<td>VTOP</td>
<td>-2147483554</td>
</tr>
<tr>
<td>VBOTTOM</td>
<td>-2147483555</td>
</tr>
</tbody>
</table>
9.17 nWidth, nHeight

Returns the width / height of the last inserted object.

property VpeCoord VPE.nWidth
property VpeCoord VPE.nHeight

    read; runtime only

Possible Values:
The width / height of the last inserted object.
9.18 VFREE

[NOT for .NET, see nFree]

A flag for indicating that VPE shall compute a coordinate dynamically. For text and images it can be used for the right coordinate (width) as well as the bottom coordinate (height). For text it means that the coordinate shall be computed due to the text-length and font size when a text object is inserted. For images the coordinate will be computed based on the resolution and dimensions found in the image file. For Rich Text (RTF) you may only set the bottom coordinate to VFREE, the right coordinate can not be dynamic.


Example:

```
VPE.Write(1, 1, VFREE, VFREE, "Hello World!")
```

Remarks:

For completeness, we list the value of VFREE here.

You should always use the named constants instead of the values.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFREE</td>
<td>-2147483549</td>
</tr>
</tbody>
</table>
9.19 nFree

[.NET and Java Only]

Instructs VPE to compute the coordinate depending on the content of the object (does only work for text, rich text and images).


**property VpeCoord VPE.nFree**

read; runtime only

**Remarks:**

This property is not available for the ActiveX / VCL.

Use the constant VFREE with the ActiveX / VCL.

**Example:**

```
Doc.Picture(1, 1, Doc.nFree, Doc.nFree, "test.bmp")
```

Instructs VPE to compute the dimensions for the right and bottom coordinate depending on the size of the image named “test.bmp”.
9.20 nLeftMargin, nTopMargin, nRightMargin, nBottomMargin

Returns / sets the specific coordinate of the Page Margins.

Note: the values for margins are specified in coordinates relative to the top / left paper border, e.g. if the right margin shall be 2cm away from the right paper border, set it to ‘page_width - 2’. If you would set the right margin = 2, it would be 2cm away from the left border.


**property VpeCoord VPE.nLeftMargin**
**property VpeCoord VPE.nTopMargin**
**property VpeCoord VPE.nRightMargin**
**property VpeCoord VPE.nBottomMargin**

read / write; runtime only

**Possible Values:**
The coordinates of the Page Margins in metric or inch units.

**Remarks:**
For completeness, we list the values of the V-Flags here.
You should always use the named constants instead of the values.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLEFTMARGIN</td>
<td>-2147483552</td>
</tr>
<tr>
<td>VRIGHTMARGIN</td>
<td>-2147483553</td>
</tr>
<tr>
<td>VTOPMARGIN</td>
<td>-2147483556</td>
</tr>
<tr>
<td>VBOTTOMMARGIN</td>
<td>-2147483557</td>
</tr>
</tbody>
</table>

If you want to use the value of a margin directly, you may use the V-Flags VLEFTMARGIN, VTOPMARGIN, VRIGHTMARGIN, VBOTTOMMARGIN instead of using the above n-properties (in case you don't want to compute offsets).
Example: VPE.Write(VLEFTMARGIN, ...) and VPE.Write(VPE.nLeftMargin, ...) are identical, but the first version is a bit faster.

But if you want to use an offset, you must use the n-properties.
Example: VPE.Write(VPE.nLeftMargin + 0.5, ...) is correct, whilst VPE.Write(VLEFTMARGIN + 0.5, ...) is not possible.

For the .NET component we always recommend to use the n-Properties instead of the V-Flags.

**See also:**
SetOutRect
9.21 SetDefOutRect

Sets the default output rectangle (= virtual Page Margins), which will be used for each new created page - NOT for the current page.

The output rectangle is very important for use with the AutoBreakMode option of VPE for rendering text.


```java
method void VPE.SetDefOutRect(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom
)
```

**Remarks:**
The method SetOutRect() calls internally SetDefOutRect() and sets the DefOutRect to the same rectangle. Therefore if you need to call SetOutRect() and SetDefOutRect() with different values at the same time, always call SetOutRect() first.

**Example:**
```
Doc.SetDefOutRect(3, 3, 19, 25)
```
9.22 SetOutRect

Sets the current output rectangle (= virtual Page Margins) for the currently active page. See "Dynamic Positioning" and "Page Margins" in the Programmer's Manual for details.

This method has the same effect like setting each nMargin property separately.

```java
method void VPE.SetOutRect(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom
)
```

VpeCoord Left, Top, Right, Bottom
the output rectangle

Remarks:
This function sets the DefOutRect to the same rectangle automatically.
9.23 StorePos
Stores the coordinates Left, Top, Right, Bottom of the last inserted object on a dynamic stack (limited in size only by available memory). The stack is a LIFO stack (Last In - First Out), which means that the last stored position is retrieved first (see RestorePos).

method void VPE.StorePos( )
9.24 RestorePos

Restores the last stored coordinates Left, Top, Right, Bottom from the stack.

method void VPE.RestorePos()
9.25 Rotation

[Not supported by the Community Edition]

Sets / returns the new rotation angle for text, images and barcodes.
Rotation is done clockwise. Angles are given in 1/10 degrees.

The x, y position will always be the same. It is not modified by rotation. Internally VPE computes the width and height of the object, and then rotates it. So rotation is easier to use, if you work with width and height (negative signs for x2 and y2) instead of absolute values.

For a detailed explanation see "Rotation of Text, Images and Barcodes" in the Programmer's Manual.

**property integer VPE.Rotation**

- read / write; runtime only
- also supported by TVPEObject (works only for TVPEObjects which reside in a template!)

**Possible Values:**
- the rotation angle in 0.1 degrees, clockwise

**Default:**
- 0 degrees

**Remarks:**
- Rotation of images is only possible, if you are using the Enhanced Edition or above. Metafiles can not be rotated.

**Example:**

```
Doc.Rotation = 900
```

The next inserted objects will be rotated by 90 degrees clockwise.
9.26 Viewable
[Professional Edition and above]
Controls, whether the next inserted object(s) are visible in the preview.

**property** boolean VPE.Viewable

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>viewable</td>
</tr>
<tr>
<td>False</td>
<td>not viewable</td>
</tr>
</tbody>
</table>

**Default:**
True
9.27 Printable

[Professional Edition and above]

Controls, whether the next inserted object(s) are printed and exported to external files, i.e. to PDF or image files.

**property boolean VPE.Printable**

  read / write; runtime only; also supported by TVPEObject

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes</td>
</tr>
<tr>
<td>False</td>
<td>no</td>
</tr>
</tbody>
</table>

**Default:**

True

**Remarks:**

By default, objects which are marked as non-printable are not exported to external file formats (for example to images or PDF). See the property `ExportNonPrintableObjects` to override this behavior.
9.28 ExportNonPrintableObjects

[Professional Edition and above]

By default, objects which are marked as non-printable are not exported to external file formats (for example to images or PDF). If this property is set to True, all objects of a document which are marked as non-printable are exported, too.

This property is document-wide in effect, it does not affect single objects.

**property boolean VPE.ExportNonPrintableObjects**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, export objects which are marked as non-printable</td>
</tr>
<tr>
<td>False</td>
<td>no, do not export objects which are marked as non-printable</td>
</tr>
</tbody>
</table>

**Default:**

False

**Remarks:**

This property does not affect VPE Document files (.VPE files). Objects which are marked as non-printable are always written to VPE Document files. The property is only relevant for export operations to external file formats, like PDF or image files.

The setting of this property is also written to VPE Document files, i.e. if you read a VPE Document file, the setting of this property is read from the file.

Example: you set the property = true and write a document to the file "test.vpe". Later you open a new document, so by default this property is false, but after you call `ReadDoc("test.vpe")`, the property will be true, because it is read from "test.vpe".

The setting of this property also affects how VPE will create PDF file attachments when sending e-mails. Furthermore it affects in the same way the document viewer VPEView.

You can set the property temporarily = True before exporting a document, and immediately afterwards back to False. This assures that it is not written to a VPE Document file.
9.29 Streamable

[Professional Edition and above]

Controls, whether the next inserted object(s) are streamed to the VPE Document File.

property boolean VPE.Streamable

read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes</td>
</tr>
<tr>
<td>False</td>
<td>no</td>
</tr>
</tbody>
</table>

Default:

True
9.30 Shadowed

[Professional Edition and above]

Draws a shadow automatically, valid for all rectangular objects.

property boolean VPE.Shadowed

read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, draw a shadow</td>
</tr>
<tr>
<td>False</td>
<td>no</td>
</tr>
</tbody>
</table>

Default:

True
9.31 ClearPage

[Professional Edition and above]

Deletes all VPE Objects of the current page. The current page will be blank and contains no objects after calling this function.

method void VPE.ClearPage(

Remarks:

  The preview will refresh automatically after calling this function.
9.32 InsertPage

[Professional Edition and above]
Inserts a new blank page at the position of the current page in the document.
The previously current page becomes the successor of the newly inserted page.
The new blank page becomes the current page.

method void VPE.InsertPage(
)

Remarks:
Pages numbered with the @PAGE macro or the $(Page) field are not renumbered after
calling this method. We recommend to use the "<page> of <total pages>" technique explained

The preview will refresh automatically after calling this function.

Enterprise Edition and above:
As you insert by code the page numbers as text objects into the document, obtain the VPE
Object handles of those text objects by using the property LastInsertedObject and store these
object references in a list. If there is a requirement for renumbering the document, first delete
all objects in this list and then start the numbering process again.
9.33 RemovePage

[Professional Edition and above]

Removes the current page from the document.
If the current page has a succeeding page, the succeeding page will become the new current page.
If there is no succeeding page (i.e. you removed the last page of the document), the preceding page will become the new current page.
If the document has only one page, it can not be removed. Instead VPE will call internally ClearPage.

method void VPE.RemovePage(
)

Remarks:
Pages numbered with the @PAGE macro or the $(Page) field are not renumbered after calling this method. We recommend to use the "<page> of <total pages>" technique explained in the Programmer's Manual.

The preview will refresh automatically after calling this function.

Enterprise Edition and above:
As you insert by code the page numbers as text objects into the document, obtain the VPE Object handles of those text objects by using the property LastInsertedObject and store these object references in a list. If there is a requirement for renumbering the document, first delete all objects in this list and then start the numbering process again.
9.34 InsertAtBottomZOrder

[Professional Edition and above]

Specifies that newly added objects will be inserted at the bottom z-order, i.e. below all previously added objects. Objects with a higher z-order are painted on top of objects with a lower z-order. This property is ideal for creating watermarks.

property boolean VPE.InsertAtBottomZOrder

read / write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>New objects are added at the bottom of the z-order (below all previously added objects)</td>
</tr>
<tr>
<td>False</td>
<td>New objects are added at the top of the z-order (above all previously added objects)</td>
</tr>
</tbody>
</table>

Default:
False
9.35 DeleteObject

[Enterprise Edition and above]

Deletes the given VPE Object from the document. References to VPE Objects can be obtained with the properties LastInsertedObject and FirstObject.

method void VPE.DeleteObject(
    TVPEObject Object
)

TVPEObject Object
    VPE Object that shall be deleted

Remarks:
    Use this method with caution: if the deleted object was inserted from a dumped template, and you try to access it with InsertedVpeObject(), an Access Violation will occur.
    As an alternative, you could make the object invisible instead of deleting it, by setting its properties Streamable, Viewable and Printable to false.
    The preview will not refresh automatically after calling this function. Call Refresh() to do so.
9.36 LastInsertedObject

[Enterprise Edition and above]

Returns the VPE Object of the last object that has been inserted into the VPE Document.

property TVPEObject VPE.LastInsertedObject

read; runtime only

Remarks:
If there is no object available, an exception is thrown.

See also:
Important Note for VPE-VCL Users in the Programmer's Manual
9.37 FirstObject

[Enterprise Edition and above]

Returns the first VPE Object of the current page. Each VPE Object itself offers the property NextObject in order to iterate through all VPE Objects of a page.

property TVPEObject VPE.FirstObject

   read; runtime only

Remarks:

   If there is no object available, the ObjectHandle of the returned object is null.

See also:

   Important Note for VPE-VCL Users in the Programmer’s Manual
10 Rendering

10.1 Rendering

These methods help to find out the size of text and images without inserting them into a document. The methods compute the size of text and images. The text or image is not inserted into the document.

Italic fonts are a bit higher than non-italic fonts. This is caused by the Windows System GDI. The consequence is, that italic text needs more height, which might result in clipped (not drawn) text, in case the height returned by a text-render method for a non-italic font is used for an italic font.

Remarks:

For framed objects (also FormFields with Bottom-Line / (Alt-)Subdivider) the center of the pen is used as basis for computations. For example if RenderWriteBox() returned as computed position / dimensions (1, 1, 2, 2) and the box of the text object has a PenSize (= frame thickness) of 1cm, then the true surrounding rectangle of the box has the coordinates (0.5, 0.5, 2.5, 2.5).

In order to compute the true outer borders you need to inflate the rectangle by ½ of the PenSize.

See also: "Important Note About Pens, Lines, Frames, Circles and Ellipses" in the Programmer's Manual.

Pictures:

Because parameters width and height can be of any value, you can for example compute the width of an undistorted image at a given height. In most cases you will set width = VFREE and height = VFREE (see "Pictures" in the Programmer's Manual).

See also:

"Rendering Objects" in the Programmer's Manual
10.2 ComputeSingleLineChars

[Not supported by the Community Edition]

Computes the maximum number of characters that fit in a single line of given width (i.e. horizontal space in a document). Only the first line of the supplied text is considered. Any subsequent lines are ignored.

ComputeSingleLineChars() accounts for the current font settings. It also allows for a left and right border line drawn at width given by the current pen size (i.e. the width passed to this method is reduced by the pen width plus a gap to account for these lines, exactly as done by PrintBox() and WriteBox()). Typically, you would set the pen size to zero and thus use the full width for the text (i.e. no border lines drawn).

This method is useful if you wish to truncate a single line of text to fit within a given width (and not have it wrap to a further line). It is also useful for more complex text rendering tasks where there is a need for piecemeal text output. However, as a rule, the other rendering methods described in this chapter are better suited and more efficient at managing the complex layout of whole text objects.

**method** integer VPE.ComputeSingleLineChars(  
    string Text,  
    VpeCoord width,  
    VslcMode [integer] mode
)

*string* Text  
the text to be rendered  

*VpeCoord* width  
the width for which the number of characters is computed  

*int* mode  
the method can be used in two different modes:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSLC_MODE_WORD</td>
<td>0</td>
<td>ModeWord</td>
<td>Truncate the line at the last complete word that fits in the width.</td>
</tr>
<tr>
<td>VSLC_MODE_CHAR</td>
<td>1</td>
<td>ModeChar</td>
<td>Truncate the line at the last character that fits in the width.</td>
</tr>
</tbody>
</table>

**Returns:**

The number of characters in the leftmost words that fit completely within the given width (mode = VSLC_MODE_WORD) or the maximum number of leftmost characters that fit completely within the given width (mode = VSLC_MODE_CHAR) based on the currently selected font and pen size.

**Remarks:**

This method only works for plain text. It does not support RTF (Rich Text). There is no equivalent method for RTF.

**Example:**
Doc.PenSize = 0
n = Doc.ComputeSingleLineChars("this is a test", 3, VSLC.MODE_WORD)

Computes which of the leftmost words in the string "this is a test" fit completely within the horizontal width of 3 cm (without border lines) using the currently selected font, and returns the number of characters (n) included in those words.

See also:

"Rendering Objects" in the Programmer's Manual
10.3 nRenderWidth, nRenderHeight
Returns the width / height of the last rendered object.

property VpeCoord VPE.nRenderWidth
property VpeCoord VPE.nRenderHeight
    read; runtime only

Possible Values:
The width / height of the last rendered object.

See also:
"Rendering Objects" in the Programmer's Manual
10.4 RenderPrint

Computes the dimensions of a given text, based on the method Print().

**method RenderStatus [integer] VPE.RenderPrint(**

VpeCoord Left,
VpeCoord Top,
string Text

**)

VpeCoord Left, Top
position
string Text
the text to be rendered

**Returns:**
The method returns one of the following values, indicating the AutoBreak status:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENDER_NO_BREAK</td>
<td>0</td>
<td>NoPageBreak</td>
<td>NO Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_BREAK</td>
<td>1</td>
<td>PageBreak</td>
<td>Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_SKIP_BREAK</td>
<td>2</td>
<td>SkipPageBreak</td>
<td>Auto Page Break will occur, but no text will be placed on the current page, all text will be skipped to the next page (nRenderWidth and nRenderHeight are not set)</td>
</tr>
</tbody>
</table>

**Remarks:**
The computed dimensions can be retrieved with the properties:
nRenderWidth and nRenderHeight

**Example:**

```pascal
Doc.RenderPrint(0, 0, "X")
font_height = Doc.nRenderHeight
```

retrieves the font height for a single line

**See also:**
"Rendering Objects" in the Programmer's Manual
10.5 RenderPrintBox

Computes the dimensions of a given text, based on the method PrintBox(). The dimensions are computed including the surrounding frame, if PenSize is <> 0.

method integer VPE.RenderPrintBox(
    VpeCoord Left,
    VpeCoord Top,
    string Text
)

VpeCoord Left, Top
position
string Text
the text to be rendered

Returns:
The method returns one of the following values, indicating the AutoBreak status:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENDER_NO_BREAK</td>
<td>0</td>
<td>NoPageBreak</td>
<td>NO Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_BREAK</td>
<td>1</td>
<td>PageBreak</td>
<td>Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_SKIP_BREAK</td>
<td>2</td>
<td>SkipPageBreak</td>
<td>Auto Page Break will occur, but no text will be placed on the current page, all text will be skipped to the next page (nRenderWidth and nRenderHeight are not set)</td>
</tr>
</tbody>
</table>

Remarks:
The computed dimensions can be retrieved with the properties:

nRenderWidth and nRenderHeight

See also:
"Rendering Objects" in the Programmer's Manual
### 10.6 RenderWrite

Computes the dimensions of a given text, based on the method `Write()`.

```pascal
method integer VPE.RenderWrite(
  VpeCoord Left,
  VpeCoord Top,
  VpeCoord Right,
  VpeCoord Bottom,
  string Text
)
```

- **VpeCoord Left, Top, Right, Bottom**
  - position and dimensions
- **string Text**
  - the text to be rendered

**Returns:**

The method returns one of the following values, indicating the AutoBreak status:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENDER_NO_BREAK</td>
<td>0</td>
<td>NoPageBreak</td>
<td>NO Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_BREAK</td>
<td>1</td>
<td>PageBreak</td>
<td>Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_SKIP_BREAK</td>
<td>2</td>
<td>SkipPageBreak</td>
<td>Auto Page Break will occur, but no text will be placed on the current page, all text will be skipped to the next page (nRenderWidth and nRenderHeight are not set)</td>
</tr>
</tbody>
</table>

**Remarks:**

The computed dimensions can be retrieved with the properties:

- `nRenderWidth` and `nRenderHeight`

**See also:**

"Rendering Objects" in the Programmer's Manual
10.7 RenderWriteBox

Computes the dimensions of a given text, based on the method `WriteBox()`. The dimensions are computed including the surrounding frame, if `PenSize` is <> 0.

**method integer VPE.RenderWriteBox(**
  VpeCoord Left,
  VpeCoord Top,
  VpeCoord Right,
  VpeCoord Bottom,
  string Text
)**

*VpeCoord Left, Top, Right, Bottom*
position and dimensions

*string Text*
the text to be rendered

**Returns:**
The method returns one of the following values, indicating the AutoBreak status:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENDER_NO_BREAK</td>
<td>0</td>
<td>NoPageBreak</td>
<td>NO Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_BREAK</td>
<td>1</td>
<td>PageBreak</td>
<td>Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_SKIP_BREAK</td>
<td>2</td>
<td>SkipPageBreak</td>
<td>Auto Page Break will occur, but no text will be placed on the current page, all text will be skipped to the next page (nRenderWidth and nRenderHeight are not set)</td>
</tr>
</tbody>
</table>

**Remarks:**
The computed dimensions can be retrieved with the properties:

`nRenderWidth` and `nRenderHeight`

**See also:**
"Rendering Objects" in the Programmer's Manual
10.8 FontAscent

[Professional Edition and above]

Returns the ascent of the currently selected font. The ascent is the height of a character from the top to the baseline. This is a text metric value.

property VpeCoord VPE.FontAscent

Returns:

  The ascent of the currently selected font.
10.9 FontDescent

[Professional Edition and above]

Returns the descent of the currently selected font. The descent is the distance from the baseline of a character to the bottom. This is a text metric value.

property VpeCoord VPE.FontDescent

Returns:

The descent of the currently selected font.
10.10 GetCharacterHeight

[Professional Edition and above]

Returns for the currently selected font the height of the bounding box of the first character in the provided string.

```csharp
method VpeCoord VPE.GetCharacterHeight(  
    string Text
)
```

**string Text**

The text to be rendered, only the first character of the string is used. Any additional characters are ignored.

**Returns:**

The method returns for the currently selected font the height of the bounding box of the first character in the provided string.

In case of an error, -1 is returned.

**Remarks:**

If you provide the capital letter "M" to this method, the returned value is the same like the text metric value "Cap Height" defined by font designers.

Using the Ascent, Descent and Character Height, the VPE API provides ways to align text at the Cap Height, the baseline or the descent of a selected font.

**Example:**

The following example draws lines at text metric positions of some text. Additionally the add-on text "Test" is drawn at the cap height position of the main text.
VpeCoord left = 2;
VpeCoord top = 2;
VpeCoord right = 18;

doc.SetFont("Arial", 72);
VpeCoord ascent = doc.FontAscent;
VpeCoord descent = doc.FontDescent;
VpeCoord cap_height = doc.CharacterHeight("M");

// top
doc.Line(left, top, right, top);

// cap height position
VpeCoord cap_pos = ascent - cap_height;
doc.PenColor = COLOR_RED;
doc.Line(left, top + cap_pos, right, top + cap_pos);

// baseline
VpeCoord baseline = top + ascent;
doc.PenColor = COLOR_BLUE;
doc.Line(left, baseline, right, baseline);

// bottom
doc.Line(left, top + ascent + descent, right, top + ascent + descent);

// text
doc.Print(left, top, "Üg My Text.");
VpeCoord x = doc.nRight;

// add-on text, positioned at cap height of main text
doc.FontSize = 26;
VpeCoord SmallCapHeight = doc.GetCharacterHeight("M");
doc.Print(x, baseline - cap_height - (doc.FontAscent - SmallCapHeight), "Test");
10.11 FontInternalLeading

[Professional Edition and above]

Returns the internal leading of the currently selected font. This value is a text metric value. It is normally not of interest and only provided for completeness of the VPE API.

property VpeCoord VPE.FontInternalLeading

Returns:

- The internal leading of the currently selected font.
10.12 FontExternalLeading

[Professional Edition and above]

Returns the external leading of the currently selected font. This value is a text metric value. It is normally not of interest and only provided for completeness of the VPE API.

property VpeCoord VPE.FontExternalLeading

Returns:

   The external leading of the currently selected font.
10.13 RenderPicture

Computes the dimensions of a given image, based on the method Picture(). The dimensions are computed including the surrounding frame, if PenSize is <> 0.

If the property PictureCache is true, the rendered image will automatically be stored in the image cache. We recommend to set PictureCache = true always.

**method void VPE.RenderPicture**(  
    VpeCoord Width,  
    VpeCoord Height,  
    string FileName  
)

*VpeCoord Width, Height*

Can be used to compute either the width or the height by setting one value to a numeric value whilst using VFREE for the other value. Usually you will set both values to VFREE.

*string FileName*

the image file to be rendered

**Remarks:**

In case of an error, LastError is set.

The computed dimensions can be retrieved with the properties:  
  nRenderWidth and nRenderHeight

**Example:**

**ActiveX / VCL:**

```vcl
Doc.PictureCache = True  
Doc.RenderPicture(VFREE, VFREE, "image.bmp")  
xsize = Doc.nRenderWidth  
ysize = Doc.nRenderHeight
```

**.NET:**

```csharp
Doc.PictureCache = True  
Doc.RenderPicture(Doc.nFree, Doc.nFree, "image.bmp")  
xsize = Doc.nRenderWidth  
ysize = Doc.nRenderHeight
```

Computes the width and height of the image stored in "image.bmp". By setting PictureCache to true, the image will be loaded into the cache, so if you insert the image later into a document, it isn’t loaded from file a second time (assuming, that it isn’t flushed from the cache, because you rendered / inserted many other - or huge - images that needed the cache). PictureCache is by default true when you open a document.

**ActiveX / VCL:**

```vcl
Doc.PictureKeepAspect = True  
Doc.RenderPicture(VFREE, 5, "image.bmp")  
xsize = Doc.nRenderWidth
```
.NET:

```csharp
Doc.PictureKeepAspect = True
Doc.RenderPicture(Doc.nFree, 5, "image.bmp")
xsize = Doc.nRenderWidth
```

Computes the undistorted width of the image stored in "image.bmp" relative to the given height of 5cm.

**See also:**

"Rendering Objects" in the Programmer's Manual
10.14 RenderPictureStream

[Professional Edition and above]

Identical to `RenderPicture()`, but renders a picture from a memory stream.

```plaintext
method void VPE.RenderPictureStream(
    TVPEStream stream,
    VpeCoord Width,
    VpeCoord Height,
    string Identifier
)
```

**TVPEStream stream**
The stream-object where the picture is read from. The stream must have been created before by calling `CreateMemoryStream()`, and of course it must have been initialized with valid image data.

**VpeCoord Width, Height**
Can be used to compute either the width or the height by setting one value to a numeric value whilst using VFREE for the other value. Usually you will set both values to VFREE.

**string Identifier**
A name for the picture. The internal image cache uses this name to distinguish images. But the image cache also computes a CRC (checksum) of the image data, so you can also leave the identifier blank (NOT NULL!). However, we do recommend to use a name if possible, so the CRC is not the only factor.

**Remarks:**
If you wish to use one and the same image multiple times, always provide the same stream handle (and therefore of course the same stream) as well as the same identifier, when calling this method.

**See also:**
"Rendering Objects" in the Programmer's Manual
10.15 RenderPictureResID

[Not available in .NET, use RenderPictureDIB]

Computes the dimensions of a given image, based on the method PictureResID(). The dimensions are computed including the surrounding frame, if PenSize is <> 0.

```csharp
method void VPE.RenderPictureResID(
    VpeCoord Width,
    VpeCoord Height,
    long hInstance,
    long ResourceID
)
```

*VpeCoord Width, Height*

Can be used to compute either the width or the height by setting one value to a numeric value whilst using VFREE for the other value. Usually you will set both values to VFREE.

*long hInstance*

Instance-Handle (of type HINSTANCE)

*long ResourceID*

the resource ID

Remarks:
In case of an error, LastError is set.

The computed dimensions can be retrieved with the properties:

*nRenderWidth and nRenderHeight*

Example:
see RenderPicture

See also:
"Rendering Objects" in the Programmer's Manual
10.16 RenderPictureResName

[Not available in .NET, use RenderPictureDIB]

Computes the dimensions of a given image, based on the method PictureResName(). The dimensions are computed including the surrounding frame, if PenSize is <> 0.

**method void VPE.RenderPictureResName(****

- **VpeCoord Width,**
- **VpeCoord Height,**
- **long hInstance,**
- **string ResourceName**

)***

**VpeCoord Width, Height**

Can be used to compute either the width or the height by setting one value to a numeric value whilst using VFREE for the other value. Usually you will set both values to VFREE.

**long hInstance**

Instance-Handle (of type HINSTANCE)

**string ResourceName**

the resource name

**Remarks:**

In case of an error, **LastError** is set.

The computed dimensions can be retrieved with the properties:

- **nRenderWidth** and **nRenderHeight**

**Example:**

see **RenderPicture**

**See also:**

"Rendering Objects" in the Programmer's Manual
10.17 RenderPictureDIB

Computes the dimensions of a given image, based on the method `PictureDIB()`. The dimensions are computed including the surrounding frame, if `PenSize` is <> 0.

```c
method void VPE.RenderPictureDIB(
    VpeCoord Width,
    VpeCoord Height,
    long hDIB
)
```

**VpeCoord Width, Height**
Can be used to compute either the width or the height by setting one value to a numeric value whilst using VFREE for the other value. Usually you will set both values to VFREE.

**long hDIB**
handle (of type HGLOBAL) to the DIB to be rendered

**Remarks:**
In case of an error, `LastError` is set.

The computed dimensions can be retrieved with the properties:

- `nRenderWidth`
- `nRenderHeight`

**Example:**
see `RenderPicture`

**See also:**
"Rendering Objects" in the Programmer's Manual
10.18 RenderRTF

[Professional Edition and above]

Computes the dimensions of a given RTF text, based on the method WriteRTF().

method integer VPE.RenderRTF(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string Text
)

VpeCoord Left, Top, Right, Bottom
position and dimensions, y2 may be set to VFREE

string Text
the RTF text to be rendered

Returns:
The method returns one of the following values, indicating the AutoBreak status:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENDER_NO_BREAK</td>
<td>0</td>
<td>NoPageBreak</td>
<td>NO Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_BREAK</td>
<td>1</td>
<td>PageBreak</td>
<td>Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_SKIP_BREAK</td>
<td>2</td>
<td>SkipPageBreak</td>
<td>Auto Page Break will occur, but no text will be placed on the current page, all text will be skipped to the next page (nRenderWidth and nRenderHeight are not set)</td>
</tr>
</tbody>
</table>

Remarks:
VFREE: only the Bottom coordinate may be set to VFREE, not the Right coordinate.
In case of an error, LastError is set.
The computed dimensions can be retrieved with the properties:
nRenderWidth and nRenderHeight

See also:
"Rendering Objects" in the Programmer's Manual
10.19 RenderBoxRTF

[Professional Edition and above]

Computes the dimensions of a given RTF text, based on the method WriteBoxRTF(). The dimensions are computed including the surrounding frame, if PenSize is <> 0.

**method integer VPE.RenderBoxRTF(**
  VpeCoord Left,
  VpeCoord Top,
  VpeCoord Right,
  VpeCoord Bottom,
  string Text
)

**VpeCoord Left, Top, Right, Bottom**
position and dimensions, y2 may be set to VFREE

**string Text**
the RTF text to be rendered

**Returns:**
The method returns one of the following values, indicating the AutoBreak status:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENDER_NO_BREAK</td>
<td>0</td>
<td>NoPageBreak</td>
<td>NO Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_BREAK</td>
<td>1</td>
<td>PageBreak</td>
<td>Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_SKIP_BREAK</td>
<td>2</td>
<td>SkipPageBreak</td>
<td>Auto Page Break will occur, but no text will be placed on the current page, all text will be skipped to the next page (nRenderWidth and nRenderHeight are not set)</td>
</tr>
</tbody>
</table>

**Remarks:**
VFREE: only the Bottom coordinate may be set to VFREE, not the Right coordinate.
In case of an error, LastError is set.
The computed dimensions can be retrieved with the properties:
nRenderWidth and nRenderHeight

**See also:**
"Rendering Objects" in the Programmer's Manual
10.20 RenderRTFFile

[Professional Edition and above]

Computes the dimensions of a given RTF text file, based on the method WriteRTFFile().

```plaintext
method integer VPE.RenderRTFFile(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string Text
)
```

*VpeCoord Left, Top, Right, Bottom*
position and dimensions, y2 may be set to VFREE

*string FileName*
the file with RTF text that is rendered

**Returns:**
The method returns one of the following values, indicating the AutoBreak status:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENDER_NO_BREAK</td>
<td>0</td>
<td>NoPageBreak</td>
<td>NO Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_BREAK</td>
<td>1</td>
<td>PageBreak</td>
<td>Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_SKIP_BREAK</td>
<td>2</td>
<td>SkipPageBreak</td>
<td>Auto Page Break will occur, but no text will be placed on the current page, all text will be skipped to the next page (nRenderWidth and nRenderHeight are not set)</td>
</tr>
</tbody>
</table>

**Remarks:**
VFREE: only the Bottom coordinate may be set to VFREE, not the Right coordinate.
In case of an error, LastError is set.
The computed dimensions can be retrieved with the properties:
nRenderWidth and nRenderHeight

**See also:**
"Rendering Objects" in the Programmer's Manual
10.21 RenderBoxRTFFile

[Professional Edition and above]

Computes the dimensions of a given RTF text file, based on the method WriteBoxRTFFile(). The dimensions are computed including the surrounding frame, if PenSize is <> 0.

**method integer VPE.RenderBoxRTFFile(**
VpeCoord Left,
VpeCoord Top,
VpeCoord Right,
VpeCoord Bottom,
string Text
**)

*VpeCoord Left, Top, Right, Bottom*
position and dimensions, y2 may be set to VFREE

*string FileName*
the file with RTF text that is rendered

**Returns:**
The method returns one of the following values, indicating the AutoBreak status:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENDER_NO_BREAK</td>
<td>0</td>
<td>NoPageBreak</td>
<td>NO Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_BREAK</td>
<td>1</td>
<td>PageBreak</td>
<td>Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_SKIP_BREAK</td>
<td>2</td>
<td>SkipPageBreak</td>
<td>Auto Page Break will occur, but no text will be placed on the current page, all text will be skipped to the next page (nRenderWidth and nRenderHeight are not set)</td>
</tr>
</tbody>
</table>

**Remarks:**
VFREE: only the Bottom coordinate may be set to VFREE, not the Right coordinate.
In case of an error, LastError is set.
The computed dimensions can be retrieved with the properties:
nRenderWidth and nRenderHeight

**See also:**
"Rendering Objects" in the Programmer's Manual
10.22 RenderRTFStream

[Professional Edition and above]
Computes the dimensions of a given RTF stream, based on the method WriteRTFFile().

**method integer VPE.RenderRTFStream(**
  TVPEStream Stream,
  VpeCoord Left,
  VpeCoord Top,
  VpeCoord Right,
  VpeCoord Bottom
**)

*TVPEStream Stream*
Stream Object

*VpeCoord Left, Top, Right, Bottom*
position and dimensions, y2 may be set to VFREE

**Returns:**
The method returns one of the following values, indicating the AutoBreak status:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENDER_NO_BREAK</td>
<td>0</td>
<td>NoPageBreak</td>
<td>NO Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_BREAK</td>
<td>1</td>
<td>PageBreak</td>
<td>Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_SKIP_BREAK</td>
<td>2</td>
<td>SkipPageBreak</td>
<td>Auto Page Break will occur, but no text will be placed on the current page, all text will be skipped to the next page (nRenderWidth and nRenderHeight are not set)</td>
</tr>
</tbody>
</table>

**Remarks:**
VFREE: only the Bottom coordinate may be set to VFREE, not the Right coordinate.

In case of an error, LastError is set.
The computed dimensions can be retrieved with the properties:
nRenderWidth and nRenderHeight

**See also:**
"Rendering Objects" in the Programmer's Manual
10.23 RenderBoxRTFStream

[Professional Edition and above]

Computes the dimensions of a given RTF stream, based on the method WriteBoxRTFFile(). The dimensions are computed including the surrounding frame, if PenSize is <> 0.

method integer VPE.RenderBoxRTFStream(
    TVPEStream Stream,
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom
)

TVPEStream Stream
Stream Object
VpeCoord Left, Top, Right, Bottom
position and dimensions, y2 may be set to VFREE

Returns:
The method returns one of the following values, indicating the AutoBreak status:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENDER_NO_BREAK</td>
<td>0</td>
<td>NoPageBreak</td>
<td>NO Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_BREAK</td>
<td>1</td>
<td>PageBreak</td>
<td>Auto Page Break will occur</td>
</tr>
<tr>
<td>RENDER_SKIP_BREAK</td>
<td>2</td>
<td>SkipPageBreak</td>
<td>Auto Page Break will occur, but no text will be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>placed on the current page, all text will be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>skipped to the next page (nRenderWidth and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>nRenderHeight are not set)</td>
</tr>
</tbody>
</table>

Remarks:
VFREE: only the Bottom coordinate may be set to VFREE, not the Right coordinate.
In case of an error, LastError is set.
The computed dimensions can be retrieved with the properties:
nRenderWidth and nRenderHeight

See also:
"Rendering Objects" in the Programmer's Manual
10.24 RenderFormField

[Enterprise Edition and above]

Computes the dimensions of a given FormField text, based on the method FormField. The dimensions are computed including the surrounding frame, if PenSize is <> 0.

```c
int VPE.RenderFormField(
    VpeCoord x,
    VpeCoord y,
    VpeCoord x2,
    VpeCoord y2,
    LPCSTR s
)
```

VpeCoord x, y, x2, y2
- position and dimensions, y2 may be set to VFREE
LPCSTR s
- the text to be rendered

Returns:
The method always returns

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>RENDER_NO_BREAK</td>
<td>0</td>
<td>NoPageBreak</td>
<td>NO Auto Page Break will occur</td>
</tr>
</tbody>
</table>

Remarks:
If x2 = VFREE, the FormField will behave the same as if VpeWrite(Box) was used.

The computed dimensions can be retrieved with the properties:
nRenderWidth and nRenderHeight

See also:
"Rendering Objects" in the Programmer's Manual
11 Drawing Functions

11.1 Drawing Functions

This section deals with the basic drawing functions, e.g. lines, polylines, polygons, boxes, ellipses, etc.; and the basic style settings for these objects, which will inherit to the text functions in the next section.
11.2 SetPen

Sets the style of the pen - all properties at once.

You can use the psXyz pen styles from Windows GDI, but pen styles other than psSolid are limited by the GDI to pens with a size of 1 pixel under Win 3.x, 9x and ME. So you should always use psSolid until the GDI changes.

method void VPE.SetPen(
    VpeCoord Size,
    PenStyle [integer] Style,
    Color Color
)

also supported by TVPEObject

VpeCoord Size
the thickness of the pen

PenStyle [integer] Style
possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>psSolid</td>
<td>0</td>
<td>Solid</td>
<td></td>
</tr>
<tr>
<td>psDash</td>
<td>1</td>
<td>Dash</td>
<td></td>
</tr>
<tr>
<td>psDot</td>
<td>2</td>
<td>Dot</td>
<td></td>
</tr>
<tr>
<td>psDashDot</td>
<td>3</td>
<td>DashDot</td>
<td></td>
</tr>
<tr>
<td>psDashDotDot</td>
<td>4</td>
<td>DashDotDot</td>
<td></td>
</tr>
</tbody>
</table>

Color Color
any of the "COLOR_xyz" constants described in Programmer's Manual
or ActiveX / VCL: any RGB value
or .NET: any member of the .NET Color structure

Default:
0.03 (which is 0.3 mm), psSolid, COLOR_BLACK
11.3 NoPen

Hides the pen (the property `PenSize` is internally set to zero). All drawing objects that are inherited from the pen-object will use no pen (see "The Object-Oriented Style" in the Programmer's Manual for details). The pen can be made visible by setting the PenSize to a different value from 0.

```java
method void VPE.NoPen()

    also supported by TVPEObject
```
11.4 PenSize

Returns / sets the pen size.

property VpeCoord VPE.PenSize

read / write; runtime only; also supported by TVPEObject

Possible Values:
the thickness of the pen

Default:
0.03 (which is 0.3 mm)

Example:

```
Doc.PenSize = 0.06    // 0.6 mm
```
11.5 PenStyle

Sets the style for the pen. You can use the psXyz pen styles from Windows GDI, but pen styles other than psSolid are limited by the GDI to pens with a size of 1 pixel under Win 3.x, 9x and ME. So you should always use psSolid until the GDI changes.

**property PenStyle [integer] VPE.PenStyle**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**
one of the windows pen styles; possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>psSolid</td>
<td>0</td>
<td>Solid</td>
<td></td>
</tr>
<tr>
<td>psDash</td>
<td>1</td>
<td>Dash</td>
<td>// ------</td>
</tr>
<tr>
<td>psDot</td>
<td>2</td>
<td>Dot</td>
<td>// ........</td>
</tr>
<tr>
<td>psDashDot</td>
<td>3</td>
<td>DashDot</td>
<td>// <em>.</em>.<em>.</em></td>
</tr>
<tr>
<td>psDashDotDot</td>
<td>4</td>
<td>DashDotDot</td>
<td>// <em>.</em>.<em>.</em></td>
</tr>
</tbody>
</table>

**Default:**
pSsolid
11.6 PenColor
Sets the color of the pen.

property Color VPE.PenColor
    read / write; runtime only; also supported by TVPEObject

Possible Values:

Color Color
    any of the "COLOR_xyz" constants described in Programmer's Manual
    or ActiveX / VCL: any RGB value
    or .NET: any member of the .NET Color structure

Default:
    COLOR_BLACK
11.7 Line

Draws a line with the current pen from the coordinate (Left, Top) to the coordinate (Right, Bottom).

```java
method void VPE.Line(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom
)
```

VpeCoord Left, Top
starting coordinates

VpeCoord Right, Bottom
ending coordinates

Remarks:

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

There is also an "Important Note About Pens, Lines, Frames, Circles and Ellipses" in the Programmer's Manual.
11.8 VpeLine

[ActiveX only]

The same as Line(). This method is for the use with Visual Basic. Visual Basic has problems with the keywords "Print, Write, Line and Scale". VB doesn't recognize, that these are methods and properties of an ActiveX.

Draws a line with the current pen from the coordinate (Left, Top) to the coordinate (Right, Bottom).

\[
\text{method void VPE.VpeLine(}
\text{  VpeCoord Left,}
\text{  VpeCoord Top,}
\text{  VpeCoord Right,}
\text{  VpeCoord Bottom}
\text{)}
\]

\text{VpeCoord Left, Top}
starting coordinates

\text{VpeCoord Right, Bottom}
ending coordinates

Remarks:
VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

There is also an: "Important Note About Pens, Lines, Frames, Circles and Ellipses" in the Programmer's Manual.
11.9 PolyLine
Creates a PolyLine object using the current pen. After a call to this method, the created PolyLine Object is ready for use with the method AddPolyPoint() (.NET: AddPoint).

**method TVPEPolyLine [pointer] VPE.PolyLine(**

```
long Count
```

**long Count**
maximum number of points (coordinates) that can be stored in this object. One point is a pair of one x and one y value.

**Returns:**

<table>
<thead>
<tr>
<th>VCL</th>
<th>a handle to the object, which can be used in further calls to AddPolyPoint()</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveX, .NET, Java, …</td>
<td>a TVPEPolyLine object, which offers the method AddPoint().</td>
</tr>
</tbody>
</table>

**Remarks:**
If you have to draw a huge number of lines at once, this method is much faster and saves memory compared to the method Line().

There is also an "Important Note About Pens, Lines, Frames, Circles and Ellipses" in the Programmer's Manual.
11.10 AddPolyPoint

[VCL only, for all other component types see the method TVPEPolyLine.AddPoint]

Adds a new point to the PolyLine object specified in hPolyLine.

method void VPE.AddPolyPoint(
    pointer hPolyLine,
    VpeCoord X,
    VpeCoord Y
)

pointer hPolyLine
    polyline object handle

VpeCoord X, Y
    coordinate of new point

- The first point you add contains the starting coordinate
- Each added point contains the next coordinate where to draw to
- If a point is -1, -1, the next coordinate is interpreted as a NEW starting coordinate
11.11 TVPEPolyLine.AddPoint
[All component types, except VCL]

Adds a new point to the PolyLine object.

method void VPE.TVPEPolyLine.AddPoint(
    VpeCoord X,
    VpeCoord Y
)

VpeCoord X, Y
coordinate of new point

- The first point you add contains the starting coordinate
- Each added point contains the next coordinate where to draw to
- If a point is -1, -1, the next coordinate is interpreted as a NEW starting coordinate
11.12 BkgMode

Sets the background mode. The background is the area inside of an object.

property BkgMode [integer] VPE.BkgMode

read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBKG_SOLID</td>
<td>0</td>
<td>Solid</td>
<td>solid background color</td>
</tr>
<tr>
<td>VBKG_TRANSPARENT</td>
<td>1</td>
<td>Transparent</td>
<td>transparent background</td>
</tr>
<tr>
<td>VBKG_GRD_LINE</td>
<td>2</td>
<td>GradientLine</td>
<td>line gradient background</td>
</tr>
<tr>
<td>VBKG_GRD_RECT</td>
<td>3</td>
<td>GradientRectangle</td>
<td>rectangular gradient background</td>
</tr>
<tr>
<td>VBKG_GRD_ELLIPSE</td>
<td>4</td>
<td>GradientEllipse</td>
<td>elliptic gradient background</td>
</tr>
</tbody>
</table>

Default:

VBKG_TRANSPARENT

Community Edition:

The Community Edition only supports VBKG_SOLID and VBKG_TRANSPARENT.
11.13 BkgColor

Sets / returns the background color. The background color is the color inside of an object. To make the background color painted, the Background Mode must be VBKG_SOLID.

**property Color VPE.BkgColor**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**
any of the "COLOR_xyz" constants described in Programmer's Manual  
**or ActiveX / VCL:** any RGB value  
**or .NET:** any member of the .NET Color structure

**Default:**
COLOR_WHITE (but the Transparent Background Mode is activated!)
11.14 BkgGradientStartColor

[Not supported by the Community Edition]

Specifies the gradient start color. To make the gradient painted, the Background Mode must be set to a gradient mode.

**property Color VPE.BkgGradientStartColor**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**
- any of the "COLOR_xyz" constants described in Programmer’s Manual
- or ActiveX / VCL: any RGB value
- or .NET: any member of the .NET Color structure

**Default:**
COLOR_WHITE (but the Transparent Background Mode is activated!)

**Remarks:**
Gradients do only work for rectangular objects like Boxes, Text, Ellipses etc. Gradients are not drawn in Pies and Polygons. If a gradient mode is selected, hatching for the same object is not possible.

**Example:**

**ActiveX / VCL:**

```pascal
Doc.BkgMode = VBKG_GRD_LINE
Doc.BkgGradientStartColor = COLOR_BLUE
Doc.BkgGradientEndColor = COLOR_GREEN
Doc.PrintBox(1, 1, "Hello World!"
```

**.NET:**

```csharp
Doc.BkgMode = BkgMode.GradientLine
Doc.BkgGradientStartColor = Color.Blue
Doc.BkgGradientEndColor = Color.Green
Doc.PrintBox(1, 1, "Hello World!"
```

Will draw the text "Hello World!" with a gradient running from blue to green in the background.
11.15 BkgGradientEndColor
[Not supported by the Community Edition]

Specifies the gradient end color. To make the gradient painted, the Background Mode must be set to a gradient mode.

property Color VPE.BkgGradientEndColor

read / write; runtime only; also supported by TVPEObject

Possible Values:
any of the "COLOR_xyz" constants described in Programmer's Manual
or ActiveX / VCL: any RGB value
or .NET: any member of the .NET Color structure

Default:
COLOR_BLACK (but the Transparent Background Mode is activated!)

Remarks:
Gradients do only work for rectangular objects like Boxes, Text, Ellipses etc. Gradients are not drawn in Pies and Polygons. If a gradient mode is selected, hatching for the same object is not possible.

Example:

ActiveX / VCL:

```vbnet
Doc.BkgMode = VBKG_GRD_LINE
Doc.BkgGradientStartColor = COLOR_BLUE
Doc.BkgGradientEndColor = COLOR_GREEN
Doc.PrintBox(1, 1, "Hello World!")
```

.NET:

```csharp
Doc.BkgMode = BkgMode.GradientLine
Doc.BkgGradientStartColor = Color.Blue
Doc.BkgGradientEndColor = Color.Green
Doc.PrintBox(1, 1, "Hello World!")
```

Will draw the text "Hello World!" with a gradient running from blue to green in the background.
11.16 BkgGradientTriColor

[Not supported by the Community Edition]

Activates / deactivates the tri-color gradient mode. If activated, you can specify a third middle color, so the gradient is drawn from the start color to the middle color and then to the end color. This allows to draw nice three-dimensional gradients.

**property bool VPE.BkgGradientTriColor**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>On</td>
</tr>
<tr>
<td>False</td>
<td>Off</td>
</tr>
</tbody>
</table>

Default: False

**Remarks:**

Tri-color gradients can only be used, when the [Background Mode](#) is set to VBKG_GRD_LINE.
11.17 BkgGradientMiddleColorPosition

[Not supported by the Community Edition]

Specifies the position of the middle color for tri-color gradients.

**property int VPE.BkgGradientMiddleColorPosition**

- read / write; runtime only; also supported by TVPEObject

**Possible Values:**

- the position in percent of the object’s height (or width, if the gradient is rotated)

**Default:**

- 50

**Remarks:**

Tri-color gradients can only be used, when the [Background Mode](#) is set to VBKG_GRD_LINE.

**Example:**

**ActiveX / VCL:**

```vcl
Doc.BkgGradientTriColor = True
Doc.BkgMode = VBKG_GRD_LINE
Doc.BkgGradientMiddleColorPosition = 35
Doc.BkgGradientStartColor = COLOR_LTGRAY
Doc.BkgGradientMiddleColor = COLOR_WHITE
Doc.BkgGradientEndColor = COLOR_DKGRAY
Doc.Print(1, 1, "Hello World!")
```

**.NET:**

```csharp
Doc.BkgGradientTriColor = True
Doc.BkgMode = BkgMode.GradientLine
Doc.BkgGradientMiddleColorPosition = 35
Doc.BkgGradientStartColor = Color.LightGray
Doc.BkgGradientMiddleColor = Color.White
Doc.BkgGradientEndColor = Color.DarkGray
Doc.Print(1, 1, "Hello World!")
```

Will draw the text "Hello World!" with a gradient running from light grey over white (at a position which is 35% of the object's height) to dark grey, which will give it a silver cylindrical appearance.
11.18 BkgGradientMiddleColor

[Not supported by the Community Edition]

Specifies the gradient middle color for tri-color gradients. To make the gradient painted, the Background Mode must be set to VBKG_GRD_LINE and the Tri Color Mode must be activated.

**property Color VPE.BkgGradientMiddleColor**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

any of the "COLOR_xyz" constants described in Programmer's Manual

or **ActiveX / VCL:** any RGB value

or **.NET:** any member of the .NET Color structure

**Default:**
COLOR_WHITE (but the Transparent Background Mode is activated and Tri Color Mode is deactivated!)

**Remarks:**
Gradients do only work for rectangular objects like Boxes, Text, Ellipses etc. Gradients are not drawn in Pies and Polygons. If a gradient mode is selected, hatching for the same object is not possible.

**Example:**

**ActiveX / VCL:**

```vbnet
Doc.BkgGradientTriColor = True
Doc.BkgMode = VBKG_GRD_LINE
Doc.BkgGradientMiddleColorPosition = 35
Doc.BkgGradientStartColor = COLOR_LTGRAY
Doc.BkgGradientMiddleColor = COLOR_WHITE
Doc.BkgGradientEndColor = COLOR_DKGRAY
Doc.Print(1, 1, "Hello World!")
```

**.NET:**

```csharp
Doc.BkgGradientTriColor = True
Doc.BkgMode = BkgMode.GradientLine
Doc.BkgGradientMiddleColorPosition = 35
Doc.BkgGradientStartColor = Color.LightGray
Doc.BkgGradientMiddleColor = Color.White
Doc.BkgGradientEndColor = Color.DarkGray
Doc.Print(1, 1, "Hello World!")
```

Will draw the text "Hello World!" with a gradient running from light grey over white (at a position which is 35% of the object’s height) to dark grey, which will give it a silver cylindrical appearance.
11.19 BkgGradientRotation

[Not supported by the Community Edition]

Specifies the rotation for the line gradient.

**property integer VPE.BkgGradientRotation**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**
rotation angle clockwise in 0.1 degrees
possible values are: 0, 900, 1800, 2700

**Default:**
0 degrees

**Example:**

**ActiveX / VCL:**

```
Doc.BkgGradientRotation = 900
Doc.BkgMode = VBKG_GRD_LINE
Doc.BkgGradientStartColor = COLOR_BLUE
Doc.BkgGradientEndColor = COLOR_GREEN
Doc.PrintBox(1, 1, "Hello World!")
```

**.NET:**

```
Doc.BkgGradientRotation = 900
Doc.BkgMode = BkgMode.GradientLine
Doc.BkgGradientStartColor = Color.Blue
Doc.BkgGradientEndColor = Color.Green
Doc.PrintBox(1, 1, "Hello World!")
```

Will draw the text "Hello World!" with a gradient rotated by 90 degrees to the right, running from blue to green in the background.
11.20 BkgGradientPrint

[Not supported by the Community Edition]

Because gradients drawn on b/w printers waste toner (or ink) and might make text and other things in the foreground unreadable, you can specify how gradients are printed. This property does not affect how gradients are drawn in the preview.

**property integer VPE.BkgGradientPrint**

write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGRD_PRINT_AUTO</td>
<td>0</td>
<td>Auto</td>
<td>if printer is a color printer, the gradient is printed, otherwise the alternate solid color is used</td>
</tr>
<tr>
<td>VGRD_PRINT_GRADIENT</td>
<td>1</td>
<td>Gradient</td>
<td>the gradient is always printed</td>
</tr>
<tr>
<td>VGRD_PRINT_SOLID</td>
<td>2</td>
<td>Solid</td>
<td>the alternate solid color is always printed</td>
</tr>
</tbody>
</table>

Default: VGRD_PRINT_AUTO

**Remarks:**

In contrast to all other graphical object properties, this property is valid for ALL objects in the document at once.

We experienced, that even some color printer drivers identify themselves as b/w printers (for example the Epson Stylus PHOTO 700 on Win-NT (driver v2.05) identifies itself as b/w printer, but the Epson Stylus Color ESC/P2 driver (shipped with Win-NT 4.0 SP 3) identifies itself as color printer).

*Printer drivers are manufactured by vendors independent of IDEAL Software; we make no warranty, implied or otherwise, regarding these product’s performance or reliability.*

If VPE should print on a color printer in VGRD_PRINT_AUTO mode gradients in b/w (so the printer driver identified itself wrongly as b/w printer), use VGRD_PRINT_GRADIENT instead.

**Example:**

**ActiveX / VCL:**

```
Doc.BkgGradientPrint = VGRD_PRINT_SOLID
Doc.BkgGradientPrintSolidColor = COLOR_LTGRAY
```

**.NET:**

```
Doc.BkgGradientPrint = BkgGradientPrint.Solid
Doc.BkgGradientPrintSolidColor = Color.LightGray
```

Will force VPE to draw a very light grey color in the background of gradient objects when they are printed.
ActiveX / VCL:

\[
\begin{align*}
\text{Doc.BkgGradientPrint} &= \text{VGRD\_PRINT\_AUTO} \\
\text{Doc.BkgGradientPrintSolidColor} &= \text{Color\_LightGray}
\end{align*}
\]

.NET:

\[
\begin{align*}
\text{Doc.BkgGradientPrint} &= \text{BkgGradientPrint\_Auto} \\
\text{Doc.BkgGradientPrintSolidColor} &= \text{Color\_LightGray}
\end{align*}
\]

VPE will check the printer during the print job, if it is a color printer. If so, the gradients are drawn. Otherwise VPE will draw a very light grey color in the background of gradient objects when they are printed.
11.21 BkgGradientPrintSolidColor

[Not supported by the Community Edition]

Specifies the alternative solid color, which shall be drawn depending on the setting for BkgGradientPrint.

property Color VPE.BkgGradientPrintSolidColor

write; runtime only

Possible Values:
- any of the "COLOR_xyz" constants described in Programmer's Manual
- or ActiveX / VCL: any RGB value
- or .NET: any member of the .NET Color structure

Default:
- COLOR_LTGRAY

Remarks:
- In contrast to all other graphical object properties, this property is valid for ALL objects in the document at once.
11.22 TransparentMode

Sets / returns the current background mode (transparent / not transparent). The background is the area inside of an object. You can use BkgMode also.

property boolean VPE.TransparentMode

    read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>on (transparent)</td>
</tr>
<tr>
<td>False</td>
<td>off (not transparent)</td>
</tr>
</tbody>
</table>

Default: True
11.23 HatchStyle

[Not supported by the Community Edition]

Sets / returns the current hatch style. All rectangular objects - except charts, barcodes and images - can be hatched with the predefined windows hatch styles.

**property long VPE.HatchStyle**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsNone</td>
<td>-1</td>
<td>None</td>
<td>see Possible styles below</td>
</tr>
<tr>
<td>hsHorizontal</td>
<td>0</td>
<td>Horizontal</td>
<td>see Possible styles below</td>
</tr>
<tr>
<td>hsVertical</td>
<td>1</td>
<td>Vertical</td>
<td>see Possible styles below</td>
</tr>
<tr>
<td>hsFDiagonal</td>
<td>2</td>
<td>ForwardDiagonal</td>
<td>see Possible styles below</td>
</tr>
<tr>
<td>hsBDiagonal</td>
<td>3</td>
<td>BackwardDiagonal</td>
<td>see Possible styles below</td>
</tr>
<tr>
<td>hsCross</td>
<td>4</td>
<td>Cross</td>
<td>see Possible styles below</td>
</tr>
<tr>
<td>hsDiagCross</td>
<td>5</td>
<td>DiagonalCross</td>
<td>see Possible styles below</td>
</tr>
</tbody>
</table>

**Default:**
hsNone, which means NO hatching

**Possible styles are:**

- HS_HORIZONTAL
- HS_EDIAGONAL
- HS_VERTICAL
- HS_CROSS
- HS_FDIAGONAL
- HS_DIAGCROSS
11.24 HatchColor

[Not supported by the Community Edition]
Sets / returns the color of the hatching.

property Color VPE.HatchColor

  read / write; runtime only; also supported by TVPEObject

Possible Values:
  any of the "COLOR_xyz" constants described in Programmer’s Manual
  or ActiveX / VCL: any RGB value
  or .NET: any member of the .NET Color structure

Default:
  COLOR_BLACK
11.25 CornerRadius

[Not supported by the Community Edition]
Sets / returns the radius of rounded corners. Setting this property different from zero causes Boxes, Text and Rich Text to be drawn with rounded corners that have the given radius.

property VpeCoord VPE.CornerRadius

    read / write; runtime only

Possible Values:
    the corner radius

Default:
    0

Remarks:
    On Win 9x/Me objects with rounded corners may not have a gradient. Gradients are drawn outside the rounded corners on Win 9x/Me.
11.26 Box

Draws a box object at position Left, Top with the right border at Right and the bottom border at Bottom. The current pen style, background mode and background color are used.

```java
method void VPE.Box(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom
)

VpeCoord Left, Top, Right, Bottom
    position and dimensions of the box
```

Remarks:
VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

There is also an "Important Note About Pens, Lines, Frames, Circles and Ellipses" in the Programmer's Manual.
11.27 Polygon

Creates a Polygon object using the current pen, background and hatchstyle. After a call to this method, the created Polygon object is ready for use with the method AddPolygonPoint() (.NET: AddPoint). See also: "Important Note About Pens, Lines, Frames, Circles and Ellipses" in the Programmer's Manual.

method TVPEPolygon [pointer] VPE.Polygon(
    long Count
)

long Count
maximum number of points (coordinates) that can be stored in this object. One point is a pair of one x and one y value.

Returns:

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCL</td>
<td>a handle to the object, which can be used in further calls to AddPolygonPoint().</td>
</tr>
<tr>
<td>ActiveX, .NET, Java, …</td>
<td>a TVPEPolygon object, which offers the method AddPoint()</td>
</tr>
</tbody>
</table>

Remarks:
There is also an "Important Note About Pens, Lines, Frames, Circles and Ellipses" in the Programmer's Manual.
11.28 AddPolygonPoint

[VCL only, for all other component types see the method TVPEPolygon.AddPoint]

Adds a new point to the Polygon object specified by the handle hPolygon.

```c
method void VPE.AddPolygonPoint(
    pointer hPolygon,
    VpeCoord X,
    VpeCoord Y
)
```

- *pointer hPolygon*
  - Polygon Object handle
- *VpeCoord x, y*
  - coordinate of new point

- The first point you add contains the starting coordinate
- Each added point contains the next coordinate where to draw to
- If a point is -1, -1, the next coordinate is interpreted as a NEW starting coordinate
11.29 TVPEPolygon.AddPoint

[All component types, except VCL]

Adds a new point to the Polygon object.

**method void VPE.TVPEPolygon.AddPoint**(
  VpeCoord X,
  VpeCoord Y
)  

_VpeCoord X, Y_

coordinate of new point

- The first point you add contains the starting coordinate
- Each added point contains the next coordinate where to draw to
- If a point is -1, -1, the next coordinate is interpreted as a NEW starting coordinate
11.30 Ellipse

Draws an ellipse or circle within the given rectangle.

Left, Top

Right, Bottom

method void VPE.Ellipse(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom
)

VpeCoord Left, Top, Right, Bottom
    surrounding rectangle of the ellipse

Remarks:
    VPE offers several methods to attach an object's position to margins and relative to the
    position of previously inserted objects. In addition Text, Rich Text and Picture objects are able
    to compute their dimensions automatically depending on their visual content.
    For details please see "Dynamic Positioning" in the Programmer's Manual.

    There is also "Important Note About Pens, Lines, Frames, Circles and Ellipses" in the
11.31 Pie

This function is identical to Ellipse(), but it draws a pie from StartAngle to EndAngle.

```java
method void VPE.Pie(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    integer StartAngle,
    integer EndAngle
)
```

**long Left, long Top, long Right, long Bottom**
surrounding rectangle of the pie

**integer BeginAngle**
angle in 0.1 degrees, clockwise

**integer EndAngle**
angle in 0.1 degrees, clockwise

**Remarks:**
VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

There is also "Important Note About Pens, Lines, Frames, Circles and Ellipses" in the Programmer's Manual.
12 Text Functions

12.1 Text Functions

These methods and properties deal with text formatting and output.
For important details about fonts and font handling, please see the Programmer's Manual, chapter "Programming Techniques", subchapter "Fonts and Font Handling".
12.2 SetFont

Selects a font and its size. The default is "Arial" and 10 pt. You can only select True-Type fonts installed on the machine on which VPE is running.

```c
method void VPE.SetFont(
    string Name,
    integer Size
)
```

also supported by TVPEObject

---

**string Name**
- font name (e.g. "Arial")

**int size**
- font size in points (not in metric or inch units!)

---

**Default:**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows and Mac OS X</td>
<td>Arial, 10pt</td>
</tr>
<tr>
<td>Any other</td>
<td>Helvetica, 10pt</td>
</tr>
</tbody>
</table>

---

**Remarks:**

VPE can only handle True-Type fonts and the 14 built-in PostScript fonts. It can not use bitmap fonts, like for example “MS Sans Serif”.

For important details about fonts and font handling, please see the Programmer's Manual, chapter "Programming Techniques", subchapter "Fonts and Font Handling".
12.3 SelectFont

The same as SetFont(). Selects a font and its size. The default is "Arial" and 10 pt. You can only select True-Type fonts installed on the machine on which VPE is running.

```csharp
method void VPE.SelectFont(
    string Name,
    integer Size
)
```

also supported by TVPEObject

- **string Name**
  - font name (e.g. "Arial")
- **int size**
  - font size in points (not in metric or inch units!)

**Default:**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows and Mac OS X</td>
<td>Arial, 10pt</td>
</tr>
<tr>
<td>Any other</td>
<td>Helvetica, 10pt</td>
</tr>
</tbody>
</table>

**Remarks:**

VPE can only handle True-Type fonts and the 14 built-in PostScript fonts. It cannot use bitmap fonts, like for example "MS Sans Serif".

For important details about fonts and font handling, please see the Programmer's Manual, chapter "Programming Techniques", subchapter "Fonts and Font Handling".
12.4 FontName

Sets a font. You can only select True-Type fonts installed on the machine on which VPE is running. Other fonts than True-Type fonts are not supported.

```plaintext
property string VPE.FontName
    read / write; runtime only; also supported by TVPEObject
```

Possible Values:
font name (e.g. "Times New Roman")

Default:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows and Mac OS X</td>
<td>Arial, 10pt</td>
</tr>
<tr>
<td>Any other</td>
<td>Helvetica, 10pt</td>
</tr>
</tbody>
</table>

Remarks:
VPE can only handle True-Type fonts and the 14 built-in PostScript fonts. It can not use bitmap fonts, like for example “MS Sans Serif”.

For important details about fonts and font handling, please see the Programmer's Manual, chapter "Programming Techniques", subchapter "Fonts and Font Handling".
12.5 FontSize
Sets a font size.

property long VPE.FontSize
   read / write; runtime only; also supported by TVPEObject

Possible Values:
   font size in points (not in metric or inch units!)

Default:
   10 pt
12.6 SetFontSubstitution

Substitutes a given font with another font. By using this method, VPE will substitute fonts immediately when creating a document. This is especially useful on Non-Windows platforms when importing RTF (Rich Text) documents. Often RTF documents are created on Windows platforms and use Windows specific True-Type fonts. With this method you can instruct VPE to substitute for example the True-Type font "Arial" with the Base 14 font "Helvetica", so Helvetica is used in place of Arial.

```
method void VPE.SetFontSubstitution(
    string OriginalFont,
    string SubstFont
)
```

- **string OriginalFont**
  - the font which shall be substituted by the SubstFont

- **string SubstFont**
  - the font which shall be used in place of the OriginalFont

**Remarks:**

In contrast to nearly all other methods, the font substitution is not related to the current document! It is active during the lifetime of your application for ALL documents which are currently open, as well as for ALL documents you are going to create after calling this method! Use this method with care.

You can reset all font substitution settings at once by calling the method `PurgeFontSubstitution()`.

On Windows, this method also affects the preview and printing. This method does not affect the document export to other formats than PDF. There is a second method `SetFontControl()` to substitute fonts, which is only active during the export to PDF documents.

For important details about fonts and font handling, please see the Programmer's Manual, chapter "Programming Techniques", subchapter "Fonts and Font Handling".

**Example:**

```
Doc.SetFontSubstitution("Arial", "Helvetica")
```

Substitutes the True-Type font "Arial" with the Base 14 font "Helvetica", so Helvetica is used in place of Arial.
12.7 PurgeFontSubstitution

Resets all font substitution settings at once. This clears all font substitution rules, so no substitution takes place.

method void VPE.PurgeFontSubstitution()
12.8 CharSet

Sets the charset.

**property CharSet [long] VPE.CharSet**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

character set, possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCHARSET_DEFAULT</td>
<td>1</td>
<td>Default</td>
<td>Windows: Chooses the VCHARSET_WIN_character set that fits to the localization settings of the current user. All other platforms: sets the charset to ISO_LATIN_1</td>
</tr>
<tr>
<td>VCHARSET_SYMBOL</td>
<td>2</td>
<td>Symbol</td>
<td>Required for using Symbol Fonts, like WingDings</td>
</tr>
<tr>
<td>VCHARSET_MAC_ROMAN</td>
<td>77</td>
<td>MacRoman</td>
<td>Macintosh Roman</td>
</tr>
</tbody>
</table>

**Character sets compatible to the character sets of the Windows operating system:**

| VCHARSET_WIN_ANSI     | 0     | WinAnsi| Western character set: Afrikaans, Basque, Catalan, Danish, Dutch, English, Esperanto, Faroese, Finnish, French, Frisian, Galician, German, Icelandic, Indonesian, Interlingua, Irish, Italian, Latin, Malay, Maltese, Norwegian, Pilipino, Portuguese, Spanish, Swahili, Swedish, Welsh |
| VCHARSET_WIN_HEBREW   | 177   | WinHebrew| Hebrew                                                                |
| VCHARSET_WIN_ARABIC   | 178   | WinArabic| Arabic                                                               |
| VCHARSET_WIN_GREEK    | 161   | WinGreek| Greek                                                                |
| VCHARSET_WIN_TURKISH  | 162   | WinTurkish| Turkish                                                              |
| VCHARSET_WIN_VIETNAMESE | 163 | WinVietnamese| Vietnamese                                                        |
| VCHARSET_WIN_THAI     | 222   | WinThai| Thai                                                                  |
| VCHARSET_WIN_EAST_EUROPE | 238 | WinEastEurope| Albanian, Belarusian, Croatian, Czech, Hungarian, Polish, Romanian, Serbian, Slovak, Slovenian |
| VCHARSET_WIN_CYRILLIC | 204   | WinCyrillic| Bulgarian, Russian, Serbian and Ukrainian (Cyrillic)                |
| VCHARSET_WIN_BALTIC   | 186   | WinBaltic| Estonian, Latvian and Lithuanian                                    |

**Character sets compatible to the ISO character sets (taken from unicode.org):**

| VCHARSET_ISO_LATIN_1  | 50    | IsoLatin1| Latin-1, Western (ISO-8859-1)                                       |
| VCHARSET_ISO_LATIN_2  | 51    | IsoLatin2| Latin-2, East European (ISO-8859-2)                                 |
| VCHARSET_ISO_LATIN_3  | 52    | IsoLatin3| Latin-3, South European (ISO-8859-3)                                |
| VCHARSET_ISO_LATIN_4  | 53    | IsoLatin4| Latin-4, Baltic (ISO-8859-4)                                        |
VCHARSET_ISO_CYRILLIC 54 IsoCyrillic Cyrillic (ISO-8859-5)
VCHARSET_ISO_ARABIC 55 IsoArabic Arabic (ISO-8859-6)
VCHARSET_ISO_GREEK 56 IsoGreek Greek (ISO-8859-7)
VCHARSET_ISO_HEBREW 57 IsoHebrew Hebrew (ISO-8859-8)
VCHARSET_ISO_LATIN_5 58 IsoLatin5 Latin-5, Turkish (ISO-8859-9)
VCHARSET_ISO_LATIN_6 59 IsoLatin6 Latin-6, Nordic (ISO-8859-10)
VCHARSET_ISO_THAI 60 isoThai Thai (ISO-8859-11)
VCHARSET_ISO_LATIN_7 62 IsoLatin7 Latin-7, Baltic (ISO-8859-13)
VCHARSET_ISO_LATIN_8 63 IsoLatin8 Latin-8, Celtic (ISO-8859-14)
VCHARSET_ISO_LATIN_9 64 IsoLatin9 Latin-9, Western (ISO-8859-15)

Default:
DEFAULT_CHARSET

Remarks:
The Mac and Iso charsets are intended for Non-Windows platforms. They should not be used with .NET, they will not function under .NET. They can be used on Windows with the ActiveX or DLL, but in this case only for PDF file creation, the preview will display wrong characters.

The supported character sets for the Base 14 fonts are limited to WinAnsi, WinEastEurope, WinTurkish, WinBaltic, IsoLatin1, IsoLatin2, IsoLatin5, IsoLatin7, IsoLatin9. MacRoman is nearly fully supported, but the following characters are missing:

- unicode 221e (ansi code 176) INFINITY
- unicode 220f (ansi code 184) GREEK CAPITAL LETTER PI
- unicode 03c0 (ansi code 185) GREEK SMALL LETTER PI
- unicode 222b (ansi code 186) INTEGRAL
- unicode 03a9 (ansi code 189) GREEK CAPITAL LETTER OMEGA
- unicode 2248 (ansi code 197) ALMOST EQUAL TO (asymptotic to)
- unicode f8ff (ansi code 240) Apple Logo.

For other character sets, True-Type fonts are required.

When exporting to PDF, some characters are missing in the WIN_ARABIC and ISO_GREEK codepages. This is due to the technique, VPE uses to define character sets within PDF documents – and that Adobe did not define those characters in their Glyphlist. A future version of VPE will overcome this problem.

The following characters of the charset WIN_ARABIC are missing:
Unicode U+06A9, Code 152, Arabic Letter "Keheh"
Unicode U+06BE, Code 170, Arabic Letter " Heh Doachashmee"
Both characters are for Persian and Urdu scripts only. So outputting arabic text is not affected.

- NOTE: the ISO_ARABIC codepage is fully supported.

The following character of the charset ISO_GREEK is missing:
Unicode U+20AF, Code 165, Drachma Sign
- NOTE: the WIN_GREEK codepage is fully supported.

To display characters from a symbol font, it is required to set the CharSet to VCHARSET_SYMBOL. VPE switches automatically the CharSet to Symbol, if a Symbol font is selected. Vice versa, VPE switches back to the previously used CharSet, if a non-Symbol font is selected.
To use a specific charset, it might be required to use a font which supports this charset. For example, if you use the font Arial with the THAI charset, this font is not available on western versions of Windows by default.

VPE does not support double-byte character sets, nor does it support UNICODE. RTL reading (right-to-left) is not supported. In a future release a UNICODE version of VPE will be created.
# 12.9 SetFontAttr

Sets all font-attributes at once.

```java
method void VPE.SetFontAttr(
    TextAlignment [integer] Alignment,
    boolean Bold,
    boolean Underlined,
    boolean Italic,
    boolean Strikeout
)
```

also supported by TVPEObject

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIGN_LEFT</td>
<td>0</td>
<td>Left</td>
<td></td>
</tr>
<tr>
<td>ALIGN_RIGHT</td>
<td>1</td>
<td>Right</td>
<td></td>
</tr>
<tr>
<td>ALIGN_CENTER</td>
<td>2</td>
<td>Center</td>
<td></td>
</tr>
<tr>
<td>ALIGN_JUSTIFIED</td>
<td>3</td>
<td>Justified</td>
<td></td>
</tr>
<tr>
<td>ALIGN_JUSTIFIED_AB</td>
<td>5</td>
<td>JustifiedAb</td>
<td></td>
</tr>
</tbody>
</table>

**boolean Bold**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>bold</td>
</tr>
<tr>
<td>False</td>
<td>not bold</td>
</tr>
</tbody>
</table>

**boolean Underlined**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>underlined</td>
</tr>
<tr>
<td>False</td>
<td>not underlined</td>
</tr>
</tbody>
</table>

**boolean Italic**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>italic</td>
</tr>
<tr>
<td>False</td>
<td>not italic</td>
</tr>
</tbody>
</table>

**boolean Strikeout**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>strikeout</td>
</tr>
<tr>
<td>False</td>
<td>not strikeout</td>
</tr>
</tbody>
</table>

**Default:**

ALIGN_LEFT, False, False, False, False
Remarks:
Italic fonts are a bit higher than non-italic fonts. This is caused by the Windows System GDI. The consequence is, that italic text needs more height, which might result in clipped (not drawn) text in case the height returned by a text-render method for a non-italic font is used for an italic font.

ALIGN_JUSTIFIED_AB: Text will be aligned justified. In contrast to ALIGN_JUSTIFIED, which aligns the last line left aligned, the last line of the text will also be aligned justified - if it is not ending with a CR / LF character. This flag works only for plain text, not for RTF.
12.10 TextAlignment
Sets the text alignment

property integer VPE.TextAlignment
   read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIGN_LEFT</td>
<td>0</td>
<td>Left</td>
<td></td>
</tr>
<tr>
<td>ALIGN_RIGHT</td>
<td>1</td>
<td>Right</td>
<td></td>
</tr>
<tr>
<td>ALIGN_CENTER</td>
<td>2</td>
<td>Center</td>
<td></td>
</tr>
<tr>
<td>ALIGN_JUSTIFIED</td>
<td>3</td>
<td>Justified</td>
<td></td>
</tr>
<tr>
<td>ALIGN_JUSTIFIED_AB</td>
<td>5</td>
<td>JustifiedAb</td>
<td></td>
</tr>
</tbody>
</table>

Default:
ALIGN_LEFT

Remarks:
ALIGN_JUSTIFIED_AB: Text will be aligned justified. In contrast to ALIGN_JUSTIFIED, which aligns the last line left aligned, the last line of the text will also be aligned justified - if it is not ending with a CR / LF character. This flag works only for plain text, not for RTF.
12.11 TextBold
Sets text bold on / off

property boolean VPE.TextBold
read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>bold</td>
</tr>
<tr>
<td>False</td>
<td>not bold</td>
</tr>
</tbody>
</table>

Default:
False
12.12 TextUnderline
Sets text underlined on / off

**property** boolean VPE.TextUnderline

read / write; runtime only; also supported by TVPEObject

### Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>underlined</td>
</tr>
<tr>
<td>False</td>
<td>not underlined</td>
</tr>
</tbody>
</table>

**Default:**
False
12.13 TextUnderlined

The same as TextUnderline. Sets text underlined on / off

**property boolean VPE.TextUnderlined**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>underlined</td>
</tr>
<tr>
<td>False</td>
<td>not underlined</td>
</tr>
</tbody>
</table>

**Default:**

False
12.14 TextItalic
Sets text italic on / off

property boolean VPE.TextItalic

    read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>italic</td>
</tr>
<tr>
<td>False</td>
<td>not italic</td>
</tr>
</tbody>
</table>

Default:
False

Remarks:
Italic fonts are a bit higher than non-italic fonts. This is caused by the Windows System GDI. The consequence is, that italic text needs more height, which might result in clipped (not drawn) text, in case the height returned by a text-render method for a non-italic font is used for an italic font.
12.15 TextStrikeOut
Sets text strikeout on / off

property boolean VPE.TextStrikeOut
   read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>strikeout</td>
</tr>
<tr>
<td>False</td>
<td>not strikeout</td>
</tr>
</tbody>
</table>

Default:
False
12.16 TextColor

Sets the text color. This is the foreground color which also applies to Barcodes, RTF and Charts.

property Color VPE.TextColor

    read / write; runtime only; also supported by TVPEObject

Possible Values:
    any of the "COLOR_xyz" constants described in Programmer's Manual
    or ActiveX / VCL: any RGB value
    or .NET: any member of the .NET Color structure

Default:
    COLOR_BLACK
12.17 Write

Outputs text formatted with the current alignment settings within a rectangle at position Left, Top with the right border at Right and the bottom border at Bottom.

The pen is invisible.

method VpeCoord VPE.Write(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string Text
)

VpeCoord Left, Top, Right, Bottom
    position and dimensions
string Text
    the text to output

Returns:
    the bottom y-coordinate generated by the output

Remarks:
    Do not use leading blanks for text output. You will realize, that this might result under special circumstances in a x- direction misalignment at the beginning of the lines, even if you are using a non-proportional font like "Courier New". This is normal and happens due to the technique, how VPE renders text. Instead of using leading blanks, eliminate those blanks and position the first character as needed by setting a correct x-start coordinate.

    You should never use static values for Y2 when doing text output, e.g. Write(1, 1, 5, 3, "Hello"). It may happen that you see text on the screen and on some printers, but not on other printers. This is, because some printer-drivers clip complete lines, in case only one pixel is outside their clipping rectangle. Also some printer-drivers add internal offsets to the bottom of text, so that it is even clipped if the text itself fits into the rectangle. Use VFREE - or for performance reasons: Render the height of a line and use this value.

    It is not possible to change font styles within one string. For such need, you should use the Professional Edition, which can interpret and display Rich Text Format (RTF).

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

    If the very first character of the output string is a [ (square bracket), VPE expects a sequence of Embedded Flags. In order to print a square bracket as the very first character, output two consecutive square brackets, i.e. [\[
. For details, see "Embedded Flag-Setting" in the Programmer's Manual.

See also:
    VpeWrite
12.18 VpeWrite

[ActiveX only]

The same as Write(). This method is for the use with Visual Basic. Visual Basic has problems with the keywords "Print, Write, Line and Scale". VB doesn't recognize, that these are methods and properties of an ActiveX.

Outputs text formatted with the current alignment settings within a rectangle at position Left, Top with the right border at Right and the bottom border at Bottom.

The pen is invisible.

```vpecoord vpecoord vpecoord vpecoord string```

VpeCoord Left, Top, Right, Bottom

position and dimensions

string Text

the text to output

Returns:

the bottom y-coordinate generated by the output

Remarks:

see also Write

It is not possible to change font styles within one string. For such need, you should use the Professional Edition, which can interpret and display Rich Text Format (RTF).

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

If the very first character of the output string is a '[' (square bracket), VPE expects a sequence of Embedded Flags. In order to print a square bracket as the very first character, output two consecutive square brackets, i.e. '[]'. For details, see "Embedded Flag-Setting" in the Programmer's Manual.
12.19 WriteBox

The same as Write(), but pen- and box-settings are used. Outputs text formatted with the current alignment settings within a rectangle at position Left, Top with the right border at Right and the bottom border at Bottom.

```vpe
method VpeCoord VPE.WriteBox(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string Text
)
```

*VpeCoord Left, Top, Right, Bottom*

position and dimensions

*string Text*

the text to output

**Returns:**
the bottom y-coordinate generated by the output

**Remarks:**
see also Write

It is not possible to change font styles within one string. For such need, you should use the Professional Edition, which can interpret and display Rich Text Format (RTF).

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

If the very first character of the output string is a '[ (square bracket), VPE expects a sequence of Embedded Flags. In order to print a square bracket as the very first character, output two consecutive square brackets, i.e. '[['. For details, see "Embedded Flag-Setting" in the Programmer's Manual.
12.20 Print

The same as Write() with the parameters (Left, Top, VFREE, VFREE).

The pen is invisible.

If the right border of the page (nRightMargin, the x2 coordinate of the Output Rectangle) is reached, the text is automatically broken to the next line; the new starting coordinate is then again x. This function does a lot of calculations and is time-consuming in relation to Write() or WriteBox() not using VFREE.

method VpeCoord VPE.Print(
    VpeCoord Left,
    VpeCoord Top,
    string Text
)

VpeCoord Left, Top
    position
string Text
    the text to output

Returns:
    the bottom y-coordinate generated by the output

Remarks:
    see also Write

    It is not possible to change font styles within one string. For such need, you should use the Professional Edition, which can interpret and display Rich Text Format (RTF).

    VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

    If the very first character of the output string is a '[' (square bracket), VPE expects a sequence of Embedded Flags. In order to print a square bracket as the very first character, output two consecutive square brackets, i.e. '[['. For details, see "Embedded Flag-Setting" in the Programmer's Manual.

See also:
    VpePrint
12.21 VpePrint

[ActiveX only]

The same as Print(). This method is for the use with Visual Basic. Visual Basic has problems with
the keywords "Print, Write, Line and Scale". VB doesn't recognize, that these are methods and
properties of an ActiveX.

The method works like Write() with the parameters (Left, Top, VFREE, VFREE).

The pen is invisible.

If the right border of the page (nRightMargin, the x2 coordinate of the Ouput Rectangle) is
reached, the text is automatically broken to the next line; the new starting coordinate is then again
x. This function does a lot of calculations and is time-consuming in relation to Write() or
WriteBox() not using VFREE.

method VpeCoord VPE.VpePrint(
    VpeCoord Left,
    VpeCoord Top,
    string Text
)

VpeCoord Left, Top
    position

string Text
    the text to output

Returns:
    the bottom y-coordinate generated by the output

Remarks:
    see also Write

    It is not possible to change font styles within one string. For such need, you should
    use the Professional Edition, which can interpret and display Rich Text Format (RTF).

    VPE offers several methods to attach an object's position to margins and relative to the
    position of previously inserted objects. In addition Text, Rich Text and Picture objects are able
    to compute their dimensions automatically depending on their visual content.
    For details please see "Dynamic Positioning" in the Programmer's Manual.

    If the very first character of the output string is a ']' (square bracket), VPE expects a sequence
    of Embedded Flags. In order to print a square bracket as the very first character, output two
    consecutive square brackets, i.e. '[]'. For details, see "Embedded Flag-Setting" in the
12.22 PrintBox

The same as WriteBox() with the parameters (Left, Top, VFREE, VFREE).

If the right border of the page (nRightMargin, the x2 coordinate of the Output Rectangle) is reached, the text is automatically broken to the next line; the new starting coordinate is then again x. This function does a lot of calculations and is time-consuming in relation to Write() or WriteBox() not using VFREE.

method VpeCoord VPE.PrintBox(
    VpeCoord Left,
    VpeCoord Top,
    string Text
)

VpeCoord Left, Top
    position
string Text
    the text to output

Returns:
    the bottom y-coordinate generated by the output

Remarks:
    see also Write

    It is not possible to change font styles within one string. For such need, you should use the Professional Edition, which can interpret and display Rich Text Format (RTF).

    VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

    If the very first character of the output string is a '[' (square bracket), VPE expects a sequence of Embedded Flags. In order to print a square bracket as the very first character, output two consecutive square brackets, i.e. '[]'. For details, see "Embedded Flag-Setting" in the Programmer's Manual.
12.23 EmbeddedFlagParser

[Not supported by the Community Edition]

Using this property, you can control whether the Embedded Flag Parser is active. When you assign the value false to this property, the Embedded Flag Parser is turned off, so text beginning with a "[" will not be interpreted as embedded flags. Instead the "]" and all following text will be inserted into the document.

**property boolean VPE.EmbeddedFlagParser**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>The Embedded Flag Parser is active.</td>
</tr>
<tr>
<td>False</td>
<td>The Embedded Flag Parser is turned off.</td>
</tr>
</tbody>
</table>

**Default:**

True = The Embedded Flag Parser is active
12.24 DefineHeader

Exactly the same as WriteBox(), but the object is inserted automatically on each new page. The string may contain the sequence "@PAGE" which will be replaced by the current page number. The header is inserted into the document by VPE on the current page and on all successively created pages. This function is intended to be used for very simple headers. For complex headers / footers see "Headers and Footers" in the Programmer's Manual.

```java
method void VPE.DefineHeader(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string Text
)
```

*VpeCoord Left, Top, Right, Bottom*
position and dimensions

*string Text*
the header text to output

**Remarks:**
After the header has been defined once, it can not be redefined nor deleted.
12.25 DefineFooter

Exactly the same as WriteBox(), but the object is inserted automatically on each new page. The string may contain the sequence "@PAGE" which will be replaced by the current page number. The footer is inserted into the document by VPE on the current page and on all successively created pages. This function is intended to be used for very simple footers. For complex headers / footers see "Headers and Footers" in the Programmer's Manual.

method void VPE.DefineFooter(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string Text
)

VpeCoord Left, Top, Right, Bottom
position and dimensions

string Text
the footer text to output

Remarks:
After the footer has been defined once, it can not be redefined nor deleted.
12.26 CharPlacement

[Professional Edition and above]

Allows to specify a constant offset from one character cell to another for text objects (not RTF). This is very useful for filling in forms that have pre-printed columns for each letter. The provided offset is understood as a "character cell width". VPE will print each character centered within this "cell".

property VpeCoord VPE.CharPlacement

write; runtime only

**Possible Values:**

the distance from one character cell to the next

**Default:**

0 (= no character placement)

**Remarks:**

When using CharPlacement, the TextAlignment must be either ALIGN_LEFT or ALIGN_RIGHT.

**Example:**

```plaintext
Doc.CharPlacement = 0.5
Doc.Print(1, 1, "123")
```

VPE will generate for each character in the given string internally a cell of 5 mm width. Each number of the string "123" will be printed centered within a cell. The first cell will be placed with the "1" at x-position 1.0 (1cm from the left paper margin), "2" at x-position 1.5 and "3" at x-position 2.0.
13 Text Block Object

[Not supported by the Community Edition]

The standard text output functions like Print() and Write() are in regular sufficient to create complex and rich documents. For special use cases VPE provides a Text Block Object, which provides more complex text layout functionality.

The Text Block Object allows to split a large block of text into smaller parts, so small pieces of the whole continuous text can be rendered and spread in any position at any width with any number of lines within a document.

As a result VPE can perform multi-column layout, text can flow around other objects or regions and text can be spread over any number of pages under full control, without using the AutoBreak event-handler.

This can be done with plain text, as well as Rich Text (RTF, requires Professional Edition or higher).
13.1 CreateTextBlock

[Not supported by the Community Edition]

Creates a Text Block Object from a given text. The current font properties are assigned to the object, i.e. font name, font size, bold, italic, underlined.

Professional Edition and higher: The font properties can be changed while fragments of the text are inserted into the document.

```csharp
method TextBlock VPE.CreateTextBlock(
    string Text
)

string Text
    the text which is assigned to the Text Block Object

Returns:
    A newly created Text Block Object.
```
13.2 CreateTextBlockRTF

[Professional Edition and higher]

Creates a Text Block Object from a given Rich Text (RTF).

**method TextBlock VPE.CreateTextBlockRTF(  
    string Text  
)  

*string Text*  
   the Rich Text which is assigned to the Text Block Object

**Returns:**  
   A newly created Text Block Object.
13.3 Delete

[Not supported by the Community Edition]

Destroys a Text Block Object and removes it from memory.

method void TextBlock.Delete()

Remarks:
A Text Block Object remains in memory, until this method is called, or when the parent document is closed. When a VPE document is closed, all Text Block Objects that belong to the document are released. It is always a good idea to call this method, if you are not using a Text Block Object anymore, in order to keep memory usage low.
13.4 HasText

[Not supported by the Community Edition]

Tests whether there is text in the supplied Text Block Object.

property boolean TextBlock.HasText

read; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>There is text in the Text Block Object</td>
</tr>
<tr>
<td>False</td>
<td>There is no text in the Text Block Object</td>
</tr>
</tbody>
</table>
13.5 Width

[Not supported by the Community Edition]

The current width of a Text Block Object. If a value is written to this property, the text within the Text Block Object is formatted accordingly to the specified width, and properties like TextBlock.Height are updated. When you call WriteTextBlock(), the text will be inserted into a document using the width you have specified with this property.

property VpeCoord TextBlock.Width

read / write; runtime only

Possible Values:
The height of a Text Block Object.

Remarks:
When a new width is assigned, VPE re-computes the internal layout of a Text Block Object.
13.6 Height

[Not supported by the Community Edition]

Returns the height of a Text Block Object, according to the width, contained text and font properties. When you output a portion of a Text Block Object, the text is removed from the Text Block Object. Therefore the height will change.

property VpeCoord TextBlock.Height

read; runtime only

Possible Values:
The height of a Text Block Object.
13.7 GetRangeHeight
[Not supported by the Community Edition]

Returns the height of a line-range of a Text Block Object, according to the width, contained text and font properties.

**method VpeCoord TextBlock.GetRangeHeight(**
    **int nFirstLine,**
    **int nLineCount,**
)

*int nFirstLine*
    the starting text line index, from which the height is computed.
    The first line has the index zero.

*int nLineCount*
    the number of text lines, for which the height is computed

**Returns:**
    The height for a range of lines of a Text Block Object.
13.8 LineCount

[Not supported by the Community Edition]

Returns the number of text lines of a Text Block Object, according to the width, contained text and font properties. When you output a portion of a Text Block Object, the text is removed from the Text Block Object. Therefore the line count will change.

property int TextBlock.LineCount

read; runtime only

Possible Values:
The number of text lines of a Text Block Object.
13.9 WriteTextBlock

[Not supported by the Community Edition]

Outputs a portion of text from the beginning of a Text Block Object, with the specified number of text lines, according to the current width and font properties.

When you output a portion of a Text Block Object, the text is removed from the Text Block Object. With successive calls to this method, you can continue to output the next part in another column or on another page.

```csharp
VpeCoord VPE.WriteTextBlock(
    TextBlock TextBlock,
    VpeCoord Left,
    VpeCoord Top,
    int nLineCount
)
```

**Parameters:**
- `TextBlock TextBlock`: Text Block Object
- `VpeCoord Left, Top`: position where the text shall be placed in the document
- `int nLineCount`: the number of text lines, which shall be placed in the document

**Returns:**
The bottom coordinate of the inserted text.

**Remarks:**
Prior to calling this function, you need to assign a value to TextBlock.Width at least once.

Assigning a value to TextBlock.Width causes to retrigger the rendering. If the font attributes (e.g. font size) for plain text are changed, you need to assign a value to TextBlock.Width to re-render the Text Block Object. This does only apply to the Professional Edition and higher. Lower editions will use the font, pen, etc. properties at the time when CreateTextBlock() is called. RTF will also ignore changes of font properties.

The AutoBreakMode, PenSize and Rotation are not in effect, neither for plain text, nor for RTF.

Differences in values returned by TextBlock.Height and TextBlock.GetRangeHeight() compared to WriteTextBlock(), Render() and other text output functions: Due to historical reasons (16-bit), Print(), Write() and Render() add a small offset to the bottom coordinate of a text object. For compatibility, WriteTextBlock() does the same. If you wish to place parts of a text block object consecutively below each other, use the values of TextBlock.Height or TextBlock.GetRangeHeight() to compute the top coordinate of the next block.

Rich Text (RTF):
Paragraph spacing properties do not work, because for each newly inserted partial text block the renderer must assume that it is the first line of the first paragraph. As a result the FirstIndent and SpaceBefore are applied to each partial text block, and SpaceBetween and SpaceAfter are ignored. As a rule of thumb, there should not be any paragraph spacing settings applied to Rich Text, which is used for Text Block Objects.

Example:
The source codes shipped with VPE contain detailed example code for many different programming languages like C#, Delphi, Java, C/C++, etc.

For using plain text with a Text Block Object, please see the 'vpedemo' source codes.

For using Rich Text (RTF) with a Text Block Object, please see the 'vppdemo' source codes.
13.10 RenderTextBlock

[Not supported by the Community Edition]

This method is useful to compute space-requirements of a Text Block Object, without inserting it into a document. Using this method, your application can make decisions regarding the layout of a Text Block, for example to issue a page break in advance.

The method renders a portion of text from the beginning of a Text Block Object, with the specified number of text lines, according to the current width and font properties.

When you render a portion of a Text Block Object, the text is removed from the Text Block Object. With successive calls to this method, you can continue to render the next part in another column or on another page.

When you have finished rendering, you can call the method Reset() to restore the whole original text you had initially supplied to the Text Block Object. This way you can re-use the Text Block Object, and call WriteTextBlock() - and all of its other methods and properties - to insert it finally into the document.

VpeCoord VPE.RenderTextBlock(
    TextBlock TextBlock,
    int nLineCount
)

TextBlock TextBlock
    Text Block Object

int nLineCount
    the number of text lines, which shall be rendered in the document

Returns:
    The bottom coordinate of the rendered text.

Remarks:
    All remarks of the method WriteTextBlock() do also apply to this method.
13.11 Reset
[Not supported by the Community Edition]
Restores the whole original text you had initially supplied to a Text Block Object.

method void TextBlock.Reset()
14 Picture Functions

14.1 Picture Functions

These methods and properties deal with the import and presentation of image files. See also "Pictures" in the Programmer's Manual for a detailed description on how to use image files.
14.2 PictureCacheSize

[Not supported by the Community Edition]

To increase performance, VPE has a build-in dynamic image cache. With this property you can set the maximum amount of memory used by VPE to cache image files.


**property long VPE.PictureCacheSize**

read / write; runtime only

**Possible Values:**
size of image cache in kilobytes (NOT megabytes)

**Default:**
65536 ( = 64 MB)

**Remarks:**
The setting of this property has great impact on performance. If you are using a lot of different images or huge images, and you know in advance that the machine has enough memory, we recommend to raise the cache size. The factory default size has been chosen very conservatively small.

Note, that the image cache does not use any memory unless an image is loaded into memory, and it uses only as much memory as required.

**Community Edition:**
The Community Edition does not provide caching of images.

**See also:**
"Pictures" in the Programmer's Manual
14.3 PictureCacheUsed
Returns the used space of the image cache in kilobytes.

property long VPE.PictureCacheUsed
    read; runtime only

Returns:
    the used space of the image cache in kilobytes

Community Edition:
    The Community Edition does not provide caching of images.

See also:
    "Pictures" in the Programmer's Manual
14.4 GetPictureTypes

Returns a string with the file name extensions of all supported file-formats.

**method string VPE.GetPictureTypes()**

**Returns:**

A list of **picture types** of the following form:

"*.WMF;*.BMP;*.TIF;*.JPG;*.PCX;*.TIF"

**Example:**

```c
string s
s = Doc.GetPictureTypes()
```

**See also:**

"Pictures" in the Programmer's Manual
14.5 PictureType

By default, VPE determines the type of an image that shall be imported or exported by its filename extension. If PictureType is other than PIC_TYPE_AUTO (the default = determining the file-type by its extension), the extension will be ignored and VPE is forced to use the specified file type. If PIC_TYPE_AUTO is used for import, VPE is able to determine the image type not only by looking at the file suffix, but also by examining the file itself. So PIC_TYPE_AUTO will work in most cases even if the file suffix does not describe the image type. Example: you try to import a JPEG image with the name "test.001" and have set PictureType = PIC_TYPE_AUTO, even then VPE will detect that this is a JPEG file!

This property applies to image-file import and export.

**property PictureType [long] VPE.PictureType**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC_TYPE_AUTO</td>
<td>255</td>
<td>Auto</td>
<td>Automatic determination by filename-suffix (default)</td>
</tr>
<tr>
<td>PIC_TYPE_BMP</td>
<td>0</td>
<td>BMP</td>
<td>Windows-Bitmap</td>
</tr>
<tr>
<td>PIC_TYPE_WMF</td>
<td>5</td>
<td>WMF</td>
<td>WMF</td>
</tr>
<tr>
<td>PIC_TYPE_EMF</td>
<td>6</td>
<td>EMF</td>
<td>EMF</td>
</tr>
<tr>
<td>PIC_TYPE_TIFF</td>
<td>64</td>
<td>TIFF</td>
<td>TIFF-File</td>
</tr>
<tr>
<td>PIC_TYPE_GIF</td>
<td>65</td>
<td>GIF</td>
<td>GIF-File</td>
</tr>
<tr>
<td>PIC_TYPE_PCX</td>
<td>66</td>
<td>PCX</td>
<td>PCX-File</td>
</tr>
<tr>
<td>PIC_TYPE_JPEG</td>
<td>68</td>
<td>JPEG</td>
<td>JPEG-File</td>
</tr>
<tr>
<td>PIC_TYPE_PNG</td>
<td>69</td>
<td>PNG</td>
<td>PNG (Portable Network Graphic)</td>
</tr>
<tr>
<td>PIC_TYPE_ICO</td>
<td>70</td>
<td>ICO</td>
<td>ICO (Windows Icon)</td>
</tr>
<tr>
<td>PIC_TYPE_JNG</td>
<td>71</td>
<td>JNG</td>
<td>JNG (JPEG Network Graphics)</td>
</tr>
<tr>
<td>PIC_TYPE_KOALA</td>
<td>72</td>
<td>KOALA</td>
<td>KOA (C64 Koala Graphics)</td>
</tr>
<tr>
<td>PIC_TYPE_IFF</td>
<td>73</td>
<td>IFF</td>
<td>IFF/LBM (Interchangeable File Format - Amiga/Deluxe Paint)</td>
</tr>
<tr>
<td>PIC_TYPE_MNG</td>
<td>74</td>
<td>MNG</td>
<td>MNG (Multiple-Image Network Graphics)</td>
</tr>
<tr>
<td>PIC_TYPE_PBM</td>
<td>75</td>
<td>PBM</td>
<td>PBM (Portable Bitmap [ASCII])</td>
</tr>
<tr>
<td>PIC_TYPE_PBM_RAW</td>
<td>76</td>
<td>PBM_RAW</td>
<td>PBM (Portable Bitmap [RAW])</td>
</tr>
<tr>
<td>PIC_TYPE_PCD</td>
<td>77</td>
<td>PCD</td>
<td>PCD (Kodak PhotoCD)</td>
</tr>
<tr>
<td>PIC_TYPE_PGM</td>
<td>78</td>
<td>PGM</td>
<td>PGM (Portable Greymap [ASCII])</td>
</tr>
<tr>
<td>PIC_TYPE_PGM_RAW</td>
<td>79</td>
<td>PGM_RAW</td>
<td>PGM (Portable Greymap [RAW])</td>
</tr>
<tr>
<td>PIC_TYPE_PPM</td>
<td>80</td>
<td>PPM</td>
<td>PPM (Portable Pixelmap [ASCII])</td>
</tr>
<tr>
<td>PIC_TYPE_PPM_RAW</td>
<td>81</td>
<td>PPM_RAW</td>
<td>PPM (Portable Pixelmap [RAW])</td>
</tr>
</tbody>
</table>
PIC_TYPE_RAS  82  RAS  RAS (Sun Raster Image)
PIC_TYPE_TARGA  83  TARGA  TGA/TARGA (Truevision Targa)
PIC_TYPE_WBMP  84  WBMP  WAP/WBMP/WBM (Wireless Bitmap)
PIC_TYPE_PSD  85  PSD  PSD (Adobe Photoshop)
PIC_TYPE_CUT  86  CUT  CUT (Dr. Halo)
PIC_TYPE_XBM  87  XBM  XBM (X11 Bitmap Format)
PIC_TYPE_XPM  88  XPM  XPM (X11 Pixmap Format)
PIC_TYPE_DDS  89  DDS  DDS (DirectX Surface)
PIC_TYPE_HDR  90  HDR  HDR (High Dynamic Range Image)
PIC_TYPE_FAX_G3  91  FAX_G3  G3 (Raw fax format CCITT G.3)
PIC_TYPE_SGI  92  SGI  SGI (SGI Image Format)

Default:
PIC_TYPE_AUTO

Remarks:
Setting this property for a VPE Object that resides in a document can have unpredictable results. In contrast you may set this property for a VPE Object that resides in a template without problems, if the file name contains at least one field.

We recommend to set this flag to PIC_TYPE_AUTO always.

Example:

ActiveX / VCL:

```
Doc.PictureType = PIC_TYPE_TIFF
Doc.Picture(1, 1, VFREE, VFREE, "image.001")
```

Will import "image.001" as TIFF file.

.NET:

```
Doc.PictureType = PictureType.TIFF
Doc.Picture(1, 1, Doc.nFree, Doc.nFree, "image.001")
```

Will import "image.001" as TIFF file.

See also:
"Pictures" in the Programmer's Manual
14.6 PictureCache

[Not supported by the Community Edition]

Instructs VPE to move all images, which are subsequently added to the document with the methods `Picture()` and `RenderPicture()`, into the dynamic image cache.


We recommend to set this property always to true.

**property boolean VPE.PictureCache**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>enabled</td>
</tr>
<tr>
<td>False</td>
<td>disabled</td>
</tr>
</tbody>
</table>

**Default:**

True

**Community Edition:**

The Community Edition does not provide caching of images.

**See also:**

"Pictures" in the Programmer's Manual
14.7 GetPicturePageCount

[Enhanced Edition and above]

Retrieves the number of available pages stored in the specified image file. At present only multipage TIFF files are supported.

```csharp
method long VPE.GetPicturePageCount(
    string FileName
)

string FileName
    image file to check for the number of available pages

Returns:
    the number of available pages stored in the specified image file

Remarks:
    In case of an error, LastError is set.

Example:

    n = Doc.GetPicturePageCount("test.tif")
```

See also:

"Pictures" in the Programmer's Manual
14.8 PicturePage

[Enhanced Edition and above]

Returns / sets the number of the page which will be loaded from a multipage image file by the next call to Picture(). The first page starts with 1.

**property long VPE.PicturePage**

read / write; runtime only

**Possible Values:**
the page number which shall be loaded from a multipage image file

**Default:**
1

**Example:**

**ActiveX / VCL:**

```plaintext
n = Doc.GetPicturePageCount("test.tif")
for i = 1 to n
    Doc.PicturePage = i
    Doc.Picture(0, 0, VFREE, VFREE, "test.tif")
    if i < n then
        Doc.PageBreak
    end if
next i
```

**.NET:**

```plaintext
n = Doc.GetPicturePageCount("test.tif")
for i = 1 to n
    Doc.PicturePage = i
    Doc.Picture(0, 0, Doc.nFree, Doc.nFree, "test.tif")
    if i < n then
        Doc.PageBreak()
    end if
next i
```

**See also:**

"Pictures" in the Programmer's Manual
14.9 PictureEmbedInDoc

When writing a VPE document to file (either by working with SwapFileName, or WriteDoc()), this property enforces that the image will be stored directly in the VPE document file with all its binary data. If this property is set to false, an image is stored with its pathname link (which will need much less space: just a few bytes) in a VPE document. Setting this property to true is useful, if the location of the referenced image changes, or if you transport a VPE document file to another system, where the image is not available (for example by e-mail).

VPE embeds images highly optimized within document files:

- Images are only stored once within a document file, regardless how often they are used (for example on each page).
- Compressed images (TIF, GIF, JPG, PNG) are stored in their original form, i.e. the images are copied directly and unchanged from the hard disk into a VPE document.
- Uncompressed images will be flatly compressed (using the ZLIB), if compression for the document file is activated (which is the default).

For details about creating and using VPE document files, please see the "Programmer's Manual", chapter "Programming Techniques", subchapter "VPE Document Files".

property boolean VPE.PictureEmbedInDoc

read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>enabled</td>
</tr>
<tr>
<td>False</td>
<td>disabled</td>
</tr>
</tbody>
</table>

Default:

True

Remarks:

ATTENTION: When you read one or more VPE document files into the current document using ReadDoc(), and those files have embedded images, you may NOT delete or modify the source document files, while the current document is open! This is, because VPE loads the embedded images directly out of the source document files each time they are not found in the image cache.

See also:

"Pictures" in the Programmer's Manual
14.10 PictureKeepAspect

Setting this mode on/off determines whether a scaled picture shall be distorted or not, when the x OR y dimension is calculated automatically. This makes sense, if only ONE parameter of a Picture-Method is VFREE: x2 OR y2. If both are VFREE, the image is always undistorted.

**property boolean VPE.PictureKeepAspect**

- write; runtime only; also supported by TVPEObject

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>on (pictures will be scaled undistorted)</td>
</tr>
<tr>
<td>False</td>
<td>Off</td>
</tr>
</tbody>
</table>

**Default:**

True

**Example:**

```plaintext
Doc.PictureKeepAspect = True
Doc.Picture(1, 1, -5, VFREE, "image1.tif")
```

In the example, the width of the image will be 5cm. If the image had an original size of 15 x 21 cm, the width is now 1/3 of the original size. Because PictureKeepAspect is True, VPE will compute the height to the same aspect ratio = 1/3 of the original height, which would then be 7 cm.

Note: the above example assumes that the resolution of the image is the same for the width and the height. Otherwise the computations performed by VPE are slightly more difficult.

**See also:**

"Pictures" in the Programmer's Manual
14.11 PictureBestFit

Keeps a picture's aspect and allows for the largest zoom within a given rectangle without distortion. In contrast to `PictureKeepAspect` this flag is useful, if you specify all four coordinates. See "Scaling" in the Programmer's Manual for details.

**property boolean VPE.PictureBestFit**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>enabled</td>
</tr>
<tr>
<td>False</td>
<td>disabled</td>
</tr>
</tbody>
</table>

**Default:**

False

**See also:**

"Pictures" in the Programmer's Manual
14.12 SetPictureDefaultDPI

Some bitmap-files do not contain the DPI resolution information in which they were originally created. For example, GIF images have no such information (you should use 96 by 96 DPI). Since VPE is a WYSIWYG-system, it needs this information for correct representation of the bitmap in metric coordinates. If the resolution information cannot be found in the image, VPE will use the default values you specify with this method.

Additionally, for the Professional Edition and above, exported bitmaps will have the resolution specified with this method.

```plaintext
method void VPE.SetPictureDefaultDPI(
    integer DPIX,
    integer DPIY
)
```

(integer DPIX, DPIY
  default x- and y-resolution in DPI

Default:
  96 by 96 DPI which is the resolution of the screen

See also:
  "Pictures" in the Programmer's Manual
14.13 PictureX2YResolution

Force the y-resolution of the image to the value of the x-resolution. Sometimes bitmaps have stored wrong resolution (DPI) information. With this flag you can instruct VPE to extract the x-resolution out of the image and to assume the y-resolution to be the same.

property boolean VPE.PictureX2YResolution

read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>enabled</td>
</tr>
<tr>
<td>False</td>
<td>disabled</td>
</tr>
</tbody>
</table>

Default:

False

See also:

"Pictures" in the Programmer's Manual
14.14 PictureDrawExact

Draws bitmaps exactly (and slow) to avoid seldom arising pixel-misalignment problems due to WYSIWYG coordinate-rounding problems, or if the graphics driver has bugs. For large bitmaps (for example full-page forms) it might happen that during scrolling a pixel line gets lost. By specifying this flag this can not happen. This flag is only useful for the preview, not for printing.

property boolean VPE.PictureEmbedInDoc

read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>enabled</td>
</tr>
<tr>
<td>False</td>
<td>disabled</td>
</tr>
</tbody>
</table>

Default: False

See also:

"Pictures" in the Programmer's Manual
14.15 PictureScale2Gray

[Professional Edition and above only]

Will perform a Scale-to-Gray of the provided bitmap for best readability at a 1:1 preview scale factor of the preview. The grayscale image is generated and than cached.

This property does not work for Metafiles, Enhanced Metafiles and DXF files.


property boolean VPE.PictureScale2Gray

    read / write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>enabled</td>
</tr>
<tr>
<td>False</td>
<td>disabled</td>
</tr>
</tbody>
</table>

Default:

False

See also:

"Pictures" in the Programmer's Manual
14.16 PictureScale2GrayFloat

[Professional Edition and above only]

Will perform a Scale-To-Gray for every preview scaling, not only for a preview scaling of 1:1 as PictureScale2Gray does.

This property does not work for Metafiles, Enhanced Metafiles and DXF files.


property boolean VPE.PictureScale2GrayFloat

read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>enabled</td>
</tr>
<tr>
<td>False</td>
<td>disabled</td>
</tr>
</tbody>
</table>

Default:
False

Remarks:
In contrast to PictureScale2Gray the **unscaled** image is cached and the grayscale image is always newly generated when displayed in a different scaling or on a new page, which needs more processor time.

See also:
"Pictures" in the Programmer’s Manual
14.17 Picture

Inserts a picture object into the document. For supported file formats, please see the chapter "Pictures" in the Programmer's Manual.

method VpeCoord VPE.Picture(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string FileName
)

VpeCoord Left, Top, Right, Bottom
position and dimensions
Dimensions: if Right is VFREE, it is computed automatically; if Bottom is VFREE, it is computed automatically. Computations are performed based on the size of the image in pixels and its resolution information in DPI (see also the properties PictureKeepAspect and PictureBestFit).

string FileName
The name of the file that shall be imported. VPE determines the file-type by the suffix (i.e. ".TIF" = TIFF, etc.) or by analyzing the file header - a full path is optional (see also: PictureType)

Returns:
the bottom coordinate of the inserted image

Remarks:
In case of an error, LastError is set.

The frame drawn around images can be made invisible by setting the PenSize to zero before importing the image.

We recommend to use the methods Picture() and PictureStream() only.
This guarantees platform independence.

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

Example:
ActiveX / VCL:
Doc.Picture(1, 1, VFREE, VFREE, "test.tif")

.NET:
Doc.Picture(1, 1, Doc.nFree, Doc.nFree, "test.tif")

See also:
"Pictures" in the Programmer's Manual
14.18 PictureStream

[Professional Edition and above]

Identical to Picture(), but imports a picture from a memory stream.

**method VpeCoord VPE.PictureStream(**

TVPEStream **stream**,
VpeCoord **Left**,
VpeCoord **Top**,
VpeCoord **Right**,
VpeCoord **Bottom**,
string **Identifier**
)

**TVPEStream stream**
The stream-object where the picture is imported from. The stream must have been created before by calling CreateMemoryStream(), and of course it must have been initialized with valid image data.

**VpeCoord Left, Top, Right, Bottom**
position and dimensions

**string Identifier**
A name for the picture. The internal image cache uses this name to distinguish images. But the image cache also computes a CRC (checksum) of the image data, so you can also leave the identifier blank (NOT NULL!). However, we do recommend to use a name if possible, so the CRC is not the only factor.

**Returns:**
the bottom coordinate of the inserted image

**Remarks:**
If you wish to use one and the same image multiple times, always provide the same stream handle (and therefore of course the same stream) as well as the same identifier, when calling this method.
14.19 PictureResID

[Not available in .NET, use PictureDIB]

Loads the image from a resource by id. The image is taken from the resource of

- the calling application, if hInstance = 0; or
- a DLL, with hInstance = handle to the instance of the loaded DLL

```c
method VpeCoord VPE.PictureResID(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    long hInstance,
    long ResourceID
)
```

`VpeCoord Left, Top, Right, Bottom`

position and dimensions

`long hInstance`

Instance-Handle (of type HINSTANCE)

`long ResourceID`

resource-id

Returns:
the bottom coordinate (y2) of the inserted image

Remarks:
From resources, VPE can only load BMP files, not JPEG, WMF, TIFF or any other file types. If you create a VPE document file with images loaded from a resource, and the PIC_IN_FILE flag is NOT used, VPE stores the resource-link in document files. If you open such a document file with another application that does not contain these resources with same ID's / names, then no images will be shown. Since DLL Instance-Handles change from load to load, VPE stores ALL images where hInstance is not null ALWAYS with the flag PIC_IN_FILE set automatically in the document files.

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

See also:
"Pictures" in the Programmer's Manual
14.20 PictureResName

[Not available in .NET, use PictureDIB]

The same as PictureResID(), but instead of a numeric ID the resource is identified by name. The image is taken from the resource of

- the calling application, if hInstance = 0; or
- a DLL, with hInstance = handle to the instance of the loaded DLL

**method VpeCoord VPE.PictureResName(**

VpeCoord Left,
VpeCoord Top,
VpeCoord Right,
VpeCoord Bottom,
long hInstance,
string ResourceName
)

VpeCoord Left, Top, Right, Bottom
position and dimensions

long hInstance
Instance-Handle (of type HINSTANCE)

string ResourceName
resource-name

**Returns:**
the bottom coordinate (y2) of the inserted image

**Remarks:**
From resources, VPE can only load BMP files, not JPEG, WMF, TIFF or any other file types. If you create a VPE document file with images loaded from a resource, and the PIC_IN_FILE flag is NOT used, VPE stores the resource-link in document files. If you open such a document file with another application that does not contain these resources with same ID’s / names, then no images will be shown. Since DLL Instance-Handles change from load to load, VPE stores ALL images where hInstance is not null ALWAYS with the flag PIC_IN_FILE set automatically in the document files.

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

**See also:**
"Pictures" in the Programmer's Manual
14.21 PictureDIB

Takes the image from a handle - you must not delete the global memory block, but keep it unlocked. The flag PIC_IN_FILE is ALWAYS automatically set. Note, that this functions expects a handle to a DIB, not to a BITMAP.

This function was only implemented to give you all possibilities in hands. Never use it to work with resources, as they are managed much better by VPE with the PictureRes-functions (locking, unlocking, conversion, etc.).

```
method VpeCoord VPE.PictureDIB(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    IntPtr [long] hDIB
)

VpeCoord Left, Top, Right, Bottom
    position and dimensions

IntPtr [long] hDIB
    handle to DIB (of type HGLOBAL)

Returns:
    the bottom coordinate (y2) of the inserted image

Remarks:
    VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content.
    For details please see "Dynamic Positioning" in the Programmer's Manual.

    PictureDIB() expects a handle for a memory block that was allocated with GlobalAlloc(). PictureDIB() creates a copy of the supplied memory block, so the calling application is responsible for freeing the DIB handle. BUT, if you supply an identical handle multiple times, VPE will not create additional copies, instead VPE will reference the initial copied memory. In other words: VPE creates only ONE copy of the supplied memory, for one and the same handle.

See also:
    "Pictures" in the Programmer's Manual
15 Barcode Functions (1D)

15.1 Barcode Functions

[Enhanced Edition and above only]

These methods and properties deal with the generation and presentation of the various barcode formats offered by VPE.

See also:
"Barcodes (1D)" in the Programmer's Manual
15.2 SetBarcodeParms

[Enhanced Edition and above]

Specifies the position of the barcode label text and the position of the add-on label text (if add-on barcode is used).

**method void VPE.SetBarcodeParms(**

- BarcodeParms [long] MainCode,
- BarcodeParms [long] AddCode

**) also supported by TVPEObject**

*BarcodeParms [long] MainCode*

one of the constants below

*BarcodeParms [long] AddCode*

one of the constants below

**Default:**

BCP_BOTTOM, BCP_BOTTOM

**Possible values for TopBottom and AddTopBottom are:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCP_BOTTOM</td>
<td>0</td>
<td>Bottom</td>
<td>Text at Bottom</td>
</tr>
<tr>
<td>BCP_TOP</td>
<td>1</td>
<td>Top</td>
<td>Text at Top</td>
</tr>
<tr>
<td>BCP_HIDE</td>
<td>2</td>
<td>Hide</td>
<td>Text Hidden</td>
</tr>
<tr>
<td>BCP_DEFAULT</td>
<td>3</td>
<td>Default</td>
<td>Text at default position appropriate to barcode type</td>
</tr>
</tbody>
</table>

**See also:**

"Barcodes (1D)" in the Programmer’s Manual
15.3 BarcodeMainTextParms
Sets the position of the main barcode label text.

property BarcodeParms [integer] VPE.BarcodeMainTextParms

read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCP_BOTTOM</td>
<td>0</td>
<td>Bottom</td>
<td>Text at Bottom</td>
</tr>
<tr>
<td>BCP_TOP</td>
<td>1</td>
<td>Top</td>
<td>Text at Top</td>
</tr>
<tr>
<td>BCP_HIDE</td>
<td>2</td>
<td>Hide</td>
<td>Text Hidden</td>
</tr>
<tr>
<td>BCP_DEFAULT</td>
<td>3</td>
<td>Default</td>
<td>Text at default position appropriate to barcode type</td>
</tr>
</tbody>
</table>

Default:
BCP_BOTTOM

See also:
"Barcodes (1D)" in the Programmer's Manual
15.4 BarcodeAddTextParms
Sets the position of the add-on barcode label text.

property BarcodeParms [integer] VPE.BarcodeMainTextParms

read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCP_BOTTOM</td>
<td>0</td>
<td>Bottom</td>
<td>Text at Bottom</td>
</tr>
<tr>
<td>BCP_TOP</td>
<td>1</td>
<td>Top</td>
<td>Text at Top</td>
</tr>
<tr>
<td>BCP_HIDE</td>
<td>2</td>
<td>Hide</td>
<td>Text Hidden</td>
</tr>
<tr>
<td>BCP_DEFAULT</td>
<td>3</td>
<td>Default</td>
<td>Text at default position appropriate to barcode type</td>
</tr>
</tbody>
</table>

Default:
BCP_BOTTOM

See also:
“Barcodes (1D)” in the Programmer’s Manual
15.5 BarcodeAlignment

[Enhanced Edition and above]
Sets the alignment of barcodes within the rectangle they are drawn.

**property BarcodeAlignment [integer] VPE.BarcodeAlignment**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIGN_LEFT</td>
<td>0</td>
<td>Left</td>
<td></td>
</tr>
<tr>
<td>ALIGN_RIGHT</td>
<td>1</td>
<td>Right</td>
<td></td>
</tr>
<tr>
<td>ALIGN_CENTER</td>
<td>2</td>
<td>Center</td>
<td></td>
</tr>
</tbody>
</table>

**Default:**
ALIGN_LEFT

**See also:**
"Barcodes (1D)" in the Programmer's Manual
15.6 BarcodeAutoChecksum

[Enhanced Edition and above]

Specifies, whether an auto checksum for a barcode will be generated automatically or not.

property integer VPE.BarcodeAutoChecksum

read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, the auto checksum will be generated</td>
</tr>
<tr>
<td>False</td>
<td>no, the auto checksum is not generated</td>
</tr>
</tbody>
</table>

Default:

True

See also:

“Barcodes (1D)” in the Programmer's Manual
15.7 BarcodeThinBar

[Enhanced Edition and above]

Specifies the relative width for thin barcode modules (1D barcodes only).

A barcode module is a line or a gap. Barcodes consist of small and thick modules.

Using BarcodeThinBar and BarcodeThickBar, the ratio of thin to thick modules can be adjusted.

**property integer VPE.BarcodeThinBar**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

the relative width for thin barcode modules, a value of zero will cause VPE to choose an optimal value in accordance to the available space for drawing the barcode

**Default:**

0

**Example:**

To adjust a ratio of 1 : 2,5 between thin and thick modules, set

```
BarcodeThinBar = 2
```

and

```
BarcodeThickBar = 5
```

**See also:**

"Barcodes (1D)" in the Programmer's Manual
15.8 BarcodeThickBar

[Enhanced Edition and above]

Specifies the relative width for thick barcode modules (1D barcodes only).
A barcode module is a line or a gap. Barcodes consist of small and thick modules.
Using BarcodeThinBar and BarcodeThickBar, the ratio of thin to thick modules can be adjusted.

property integer VPE.BarcodeThickBar

read / write; runtime only; also supported by TVPEObject

Possible Values:
the relative width for thick barcode modules, a value of zero will cause VPE to choose an
optimal value in accordance to the available space for drawing the barcode

Default:
0

Example:
To adjust a ratio of 1 : 2,5 between thin and thick modules, set

BarcodeThinBar = 2

and

BarcodeThickBar = 5

See also:
"Barcodes (1D)" in the Programmer's Manual
15.9 Barcode

[Enhanced Edition and above]

Generates and draws a barcode within a rectangle at position Left, Top with the right border at Right and the bottom border at Bottom.

VPE doesn't enforce the absolute size of the barcode (left to the responsibility of the caller), but it does enforce the exactness of the relative widths of the bars at the output device's pixel level (other than screen).

Consider a barcode consisting of 1 bar, 1 space and 1 bar, with width ratios 3:1:2. The minimum width of the barcode is 6 pixels, other possible sizes are 12, 18 and so on. If you give a rectangle where only 5 pixels might fit, the barcode will still occupy 6 pixels. Don't make the rectangle too small. Normally the barcodes will be drawn inside the rectangle and as big as possible due to the exactness of the relative widths of the bars.

The color of bars and barcode text is set by TextColor. The font used for the barcode labels can be specified with the property FontName, for example UPC-A and UPC-E require an OCR-B font. In addition the font size for barcode labels can be specified with the property FontSize.

**method void VPE.Barcode(**

  VpeCoord Left,
  VpeCoord Top,
  VpeCoord Right,
  VpeCoord Bottom,
  BarcodeType [integer] CodeType,
  string Code,
  string AddCode

  )

VpeCoord Left, Top, Right, Bottom
position and dimensions

BarcodeType [integer] CodeType
see below

string Code
  string with the code (e.g. "123456")

string AddCode
  string with the add-on code, if add-on barcode type chosen, else "" (empty string or NULL (0))

the parameter "CodeType" can be one of the following values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCT_EAN13</td>
<td>1</td>
<td>Ean13</td>
<td>Checkdigit required, OCR-B Font required</td>
</tr>
<tr>
<td>BCT_EAN13_2</td>
<td>10</td>
<td>Ean13_2</td>
<td>Checkdigit required, OCR-B Font required</td>
</tr>
<tr>
<td>BCT_EAN13_5</td>
<td>11</td>
<td>Ean13_5</td>
<td>Checkdigit required, OCR-B Font required</td>
</tr>
<tr>
<td>BCT_EAN8</td>
<td>2</td>
<td>Ean8</td>
<td>Checkdigit required, OCR-B Font required</td>
</tr>
<tr>
<td>BCT_EAN8_2</td>
<td>12</td>
<td>Ean8_2</td>
<td>Checkdigit required, OCR-B Font required</td>
</tr>
<tr>
<td>BCT_EAN8_5</td>
<td>13</td>
<td>Ean8_5</td>
<td>Checkdigit required, OCR-B Font required</td>
</tr>
<tr>
<td>BCT_UPCA</td>
<td>3</td>
<td>UpcA</td>
<td>Checkdigit required, OCR-B Font required</td>
</tr>
<tr>
<td>BCT_UPCA_2</td>
<td>14</td>
<td>UpcA_2</td>
<td>Checkdigit required, OCR-B Font required</td>
</tr>
<tr>
<td>Barcode Function</td>
<td>Parameter</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>BCT_UPCA_5</td>
<td></td>
<td>Checkdigit required, OCR-B Font required</td>
<td></td>
</tr>
<tr>
<td>BCT_UPCE</td>
<td>9</td>
<td>UpcE, first digit always '0', Checkdigit required, OCR-B Font required</td>
<td></td>
</tr>
<tr>
<td>BCT_UPCE_2</td>
<td>16</td>
<td>UpcE_2, first digit always '0', Checkdigit required, OCR-B Font required</td>
<td></td>
</tr>
<tr>
<td>BCT_UPCE_5</td>
<td>17</td>
<td>UpcE_5, first digit always '0', Checkdigit required, OCR-B Font required</td>
<td></td>
</tr>
<tr>
<td>BCT_EAN2</td>
<td>28</td>
<td>Ean2, no Checkdigit</td>
<td></td>
</tr>
<tr>
<td>BCT_EAN5</td>
<td>29</td>
<td>Ean5, no Checkdigit</td>
<td></td>
</tr>
<tr>
<td>BCT_CODABAR</td>
<td>5</td>
<td>Codabar, Checkdigit optional, uThick / uThin</td>
<td></td>
</tr>
<tr>
<td>BCT_2OF5</td>
<td>7</td>
<td>B2Of5, 2 of 5 Industrial (25ID), Checkdigit optional, uThick / uThin</td>
<td></td>
</tr>
<tr>
<td>BCT_INTERLEAVED2OF5</td>
<td>8</td>
<td>Interleaved2Of5, 2 of 5 Interleaved (25IL), Checkdigit optional, uThick / uThin</td>
<td></td>
</tr>
<tr>
<td>BCT_CODE39</td>
<td>6</td>
<td>Code39, Checkdigit optional, uThick / uThin</td>
<td></td>
</tr>
<tr>
<td>BCT_CODE39EXT</td>
<td>30</td>
<td>Code39Ext, Code 3 of 9 Extended, Checkdigit optional, uThick / uThin</td>
<td></td>
</tr>
<tr>
<td>BCT_CODE93</td>
<td>21</td>
<td>Code93, Checkdigit optional</td>
<td></td>
</tr>
<tr>
<td>BCT_CODE93EXT</td>
<td>31</td>
<td>Code93Ext, Code 93 Extended, Checkdigit optional</td>
<td></td>
</tr>
<tr>
<td>BCT_POSTNET</td>
<td>22</td>
<td>Postnet, uThick / uThin are fixed</td>
<td></td>
</tr>
<tr>
<td>BCT_EAN128</td>
<td>27</td>
<td>Ean128, set of EAN 128A, B and C, Checkdigit required</td>
<td></td>
</tr>
<tr>
<td>BCT_RM4SCC</td>
<td>32</td>
<td>Rm4Scc, Royal Mail 4 State Customer Code, Checkdigit required</td>
<td></td>
</tr>
<tr>
<td>BCT_MSI</td>
<td>33</td>
<td>Msi, Checkdigit optional, uThick / uThin</td>
<td></td>
</tr>
<tr>
<td>BCT_IDENTCODE</td>
<td>36</td>
<td>Identcode, Identcode of the Deutsche Post AG, Checkdigit required</td>
<td></td>
</tr>
<tr>
<td>BCT_LEITCODE</td>
<td>37</td>
<td>Leitcode, Leitcode of the Deutsche Post AG, Checkdigit required</td>
<td></td>
</tr>
<tr>
<td>BCT_PZN</td>
<td>38</td>
<td>Pzn, Pharma Zentral Code, Checkdigit required, uThick / uThin</td>
<td></td>
</tr>
<tr>
<td>BCT_CODE11</td>
<td>39</td>
<td>Code11, Checkdigit optional, uThick / uThin</td>
<td></td>
</tr>
<tr>
<td>BCT_2OF5MATRIX</td>
<td>40</td>
<td>B2OF5Matrix, 2 of 5 Matrix, Checkdigit optional, uThick / uThin</td>
<td></td>
</tr>
<tr>
<td>BCT_TELEPENA</td>
<td>41</td>
<td>TelepenA, Checkdigit optional</td>
<td></td>
</tr>
<tr>
<td>BCT_INTELLIGENT_MAIL</td>
<td>42</td>
<td>IntelligentMail, Intelligent Mail</td>
<td></td>
</tr>
</tbody>
</table>

**Remarks:**

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see “Dynamic Positioning” in the Programmer's Manual.

*If you are using the ActiveX, do not supply the value 0 (zero) for the AddCode parameter, if the barcode type does not support add-on codes. Instead, supply an empty string, i.e. "".*

**Example:**

```plaintext
Doc.Barcode 1, 1, -4, -1, BCT_INTERLEAVED2OF5, "12345", ""
```
16 Barcode Functions (2D)

16.1 Barcode Functions (2D)

[Professional Edition and above only]

These methods and properties deal with the generation and presentation of the various two-dimensional barcodes offered by VPE.

See also:
"Barcodes (2D)" in the Programmer's Manual
16.2 Bar2DAlignment

[Professional Edition and above only]
Sets the horizontal and vertical alignment of the 2D-barcodes within the given rectangle.

**property Bar2DAlignment [integer] VPE.Bar2DAlignment**
read / write; runtime only; also supported by TVPEObject

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBAR2D_ALIGN_CENTER</td>
<td>0</td>
<td>Center</td>
<td>horizontally and vertically centered</td>
</tr>
<tr>
<td>VBAR2D_ALIGN_CENTER_H</td>
<td>0</td>
<td>CenterH</td>
<td>horizontally centered</td>
</tr>
<tr>
<td>VBAR2D_ALIGN_CENTER_V</td>
<td>0</td>
<td>CenterV</td>
<td>vertically centered</td>
</tr>
<tr>
<td>VBAR2D_ALIGN_LEFT</td>
<td>1</td>
<td>Left</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_ALIGN_RIGHT</td>
<td>2</td>
<td>Right</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_ALIGN_TOP</td>
<td>4</td>
<td>Top</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_ALIGN_BOTTOM</td>
<td>8</td>
<td>Bottom</td>
<td></td>
</tr>
</tbody>
</table>

**Default:**
VBAR2D_ALIGN_CENTER

**Remarks:**
You combine values for horizontal and vertical alignment by adding two flags.

**Example:**

ActiveX / VCL:

```
Doc.Bar2DAlignment = VBAR2D_ALIGN_LEFT + VBAR2D_ALIGN_TOP
```

Will align the barcode to the top-left corner.

.NET:

```
VB: Doc.Bar2DAlignment = Bar2DAlignment.Left + Bar2DAlignment.Top
C#: Doc.Bar2DAlignment = Bar2DAlignment.Left | Bar2DAlignment.Top
```

Will align the barcode to the top-left corner.

**See also:**
"Barcodes (2D)" in the Programmer's Manual
16.3 DataMatrixEncodingFormat

[Professional Edition and above only]

Sets the encoding format ID for DataMatrix symbologies.

property DataMatrixEncodingFormat [integer] VPE.DataMatrixEncodingFormat

read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBAR2D_DATA_MATRIX_ENC_BASE11</td>
<td>1</td>
<td>Base11</td>
<td>numeric data, 3.5 bits per digit</td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ENC_BASE27</td>
<td>2</td>
<td>Base27</td>
<td>upper case alphabetic, 4.8 bits per character</td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ENC_BASE41</td>
<td>3</td>
<td>Base41</td>
<td>upper case alphanumeric and punctuation, 5.5 bits per character</td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ENC_BASE37</td>
<td>4</td>
<td>Base37</td>
<td>upper case alphanumeric, 5.25 bits per character</td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ENC_ASCII</td>
<td>5</td>
<td>Ascii</td>
<td>128 ASCII set, 7 bits per character</td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ENC_BINARY</td>
<td>6</td>
<td>Binary</td>
<td>8-bit Byte, 8 bits per character</td>
</tr>
</tbody>
</table>

Default:
VBAR2D_DATA_MATRIX_ENC_BASE11

Remarks:
For ECC200 this is the start-up code page, while for ECC000 - ECC140 the selected format covers the entire message.

See also:
"Barcodes (2D)" in the Programmer's Manual
16.4 DataMatrixEccType

[Professional Edition and above only]

Sets the error checking and correction scheme for DataMatrix symbologies.

property DataMatrixEccType [integer] VPE.DataMatrixEccType

read / write; runtime only; also supported by TVPEObject

Possible Values:
the error correction scheme:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC000</td>
<td>0</td>
<td>Ecc000</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC010</td>
<td>1</td>
<td>Ecc010</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC040</td>
<td>2</td>
<td>Ecc040</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC050</td>
<td>3</td>
<td>Ecc050</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC060</td>
<td>4</td>
<td>Ecc060</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC070</td>
<td>5</td>
<td>Ecc070</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC080</td>
<td>6</td>
<td>Ecc080</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC090</td>
<td>7</td>
<td>Ecc090</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC100</td>
<td>8</td>
<td>Ecc100</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC110</td>
<td>9</td>
<td>Ecc110</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC120</td>
<td>10</td>
<td>Ecc120</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC130</td>
<td>11</td>
<td>Ecc130</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC140</td>
<td>12</td>
<td>Ecc140</td>
<td></td>
</tr>
<tr>
<td>VBAR2D_DATA_MATRIX_ECC200</td>
<td>26</td>
<td>Ecc200</td>
<td></td>
</tr>
</tbody>
</table>

Default:
VBAR2D_DATA_MATRIX_ECC200

See also:
"Barcodes (2D)" in the Programmer's Manual
16.5 DataMatrixRows

[Professional Edition and above only]

Sets the desired number of matrix rows (0: choose automatically based on message).

property integer VPE.DataMatrixRows

read / write; runtime only; also supported by TVPEObject

Possible Values:
Desired number of matrix rows (0: choose automatically based on message).

Default:
0

Remarks:
Note: The number of columns and rows heavily depends on the ECC type that has been selected. For example, a valid ECC100 symbol spans at least 13x13 modules and at most 49x49, where only odd combinations are allowed (i.e. 13x13, 15x15, 17x17, .. 47x47, 49x49).

See also:
"Barcodes (2D)" in the Programmer's Manual
16.6 DataMatrixColumns

[Professional Edition and above only]

Sets the desired number of matrix columns (0: choose automatically based on message).

**property integer VPE.DataMatrixColumns**

   read / write; runtime only; also supported by TVPEObject

**Possible Values:**

   Desired number of matrix columns (0: choose automatically based on message).

**Default:**

   0

**Remarks:**

   Note: The number of columns and rows heavily depends on the ECC type that has been selected. For example, a valid ECC100 symbol spans at least 13x13 modules and at most 49x49, where only odd combinations are allowed (i.e. 13x13, 15x15, 17x17, .. 47x47, 49x49).

**See also:**

   "Barcodes (2D)" in the Programmer's Manual
16.7 DataMatrixMirror

[Professional Edition and above only]

Specifies, whether a mirrored version of the barcode shall be created (applies to ECC000 – ECC140 only).

property boolean VPE.DataMatrixMirror

    read / write; runtime only; also supported by TVPEObject

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>do not mirror</td>
</tr>
<tr>
<td>False</td>
<td>Mirror</td>
</tr>
</tbody>
</table>

Default:

    False

See also:

"Barcodes (2D)" in the Programmer's Manual
16.8 DataMatrixBorder

[Professional Edition and above only]
Sets the number of modules used for the border (usually 1).

property integer VPE.DataMatrixBorder

   read / write; runtime only; also supported by TVPEObject

Possible Values:
   the number of modules used for the border (usually 1)

Default:
   1

See also:
   "Barcodes (2D)" in the Programmer's Manual
16.9 DataMatrix

[Professional Edition and above only]

Inserts a DataMatrix barcode into the VPE document.

**method void VPE.DataMatrix(**
  VpeCoord *Left*,
  VpeCoord *Top*,
  VpeCoord *Right*,
  VpeCoord *Bottom*,
  string *Text*
)

*VpeCoord* *Left, Top, Right, Bottom* coordinates

*string Text* the text of the barcode

**Example:**

```
Doc.DataMatrix(1, 1, -3, -3, "30Q32434340794<QQQ")
```

**Remarks:**

sets *LastError*

**See also:**

"Barcodes (2D)" in the Programmer's Manual
16.10 RenderDataMatrix

[Professional Edition and above only]

Computes the dimensions of a DataMatrix barcode.

**method void VPE.RenderDataMatrix(**
```
   string Text,
   VpeCoord nWidth,
   VpeCoord nHeight,
   VpeCoord nModuleWidth
```
**)

*string Text*
   the text of the barcode

*VpeCoord nWidth, nHeight*
   With these two parameters you can specify a maximum size the barcode can have.
   The following rules apply:
   
   - nWidth and nHeight is specified: in this case the maximum rectangle of the barcode
     will be computed, which will fit into the given rectangle.
   - Only nWidth is specified, nHeight is zero: In this case nHeight is computed
     (nWidth is also computed, i.e. adjusted).
   - Only nHeight is specified, nWidth is zero: In this case nWidth is computed
     (nHeight is also computed, i.e. adjusted).
   - nWidth and nHeight are zero: in this case the smallest possible rectangle is
     computed

*VpeCoord nModuleWidth*
   If this parameter is zero, VPE will choose itself an optimum barcode module width. If it is non-
   zero, you can specify the width of a barcode module.

**Example:**
```
// Compute the smallest possible rectangle for a given text
Doc.RenderDataMatrix("30Q324343430794<OQQ", 0, 0, 0)
xsise = Doc.nRenderWidth
ysize = Doc.nRenderHeight

// Insert the barcode into the document
Doc.DataMatrix(1, 1, -xsize, -ysize, "30Q324343430794<OQQ")
```

**Remarks:**
   sets LastError

**See also:**
   "Barcodes (2D)" in the Programmer's Manual
16.11 QRCodeVersion

[Professional Edition and above only]

Defines the maximum data capacity for QR Codes (depending on EccLevel and encoding Mode), see http://www.qrcode.com.

The symbol versions of QR Code range from Version 1 to Version 40. Each version has a different module configuration or number of modules. (The module refers to the black and white dots that make up QR Code.)

"Module configuration" refers to the number of modules contained in a symbol, commencing with Version 1 (21 × 21 modules) up to Version 40 (177 × 177 modules). Each higher version number comprises 4 additional modules per side.

property integer VPE.QRCodeVersion

    read / write; runtime only; also supported by TVPEObject

Possible Values:

    the version, 0 = automatic

Default:

    0 = automatic

See also:

    "Barcodes (2D)" in the Programmer's Manual
16.12 QRCodeEccLevel

[Professional Edition and above only]

Sets the error correction code level for QR Codes. The higher the level, i.e. the higher the error
tolerance, the less user data can be encoded within a symbol.

**property QRCodeEccLevel [integer] VPE.QRCodeEccLevel**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**
the error correction code level

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBAR2D_QRCODE_ECC_LEVEL_L</td>
<td>0</td>
<td>L</td>
<td>7% of codewords can be restored</td>
</tr>
<tr>
<td>VBAR2D_QRCODE_ECC_LEVEL_M</td>
<td>1</td>
<td>M</td>
<td>15% of codewords can be restored</td>
</tr>
<tr>
<td>VBAR2D_QRCODE_ECC_LEVEL_Q</td>
<td>2</td>
<td>Q</td>
<td>25% of codewords can be restored</td>
</tr>
<tr>
<td>VBAR2D_QRCODE_ECC_LEVEL_H</td>
<td>3</td>
<td>H</td>
<td>30% of codewords can be restored</td>
</tr>
</tbody>
</table>

**Default:**
0 = VBAR2D_QRCODE_ECC_LEVEL_L

**See also:**
"Barcodes (2D)" in the Programmer's Manual
16.13 QRCodeMode

[Professional Edition and above only]

Sets encoding mode for QR Codes. The QR Code provides four different encoding modes, the user data capacity depends on the encoding and error correction level:

<table>
<thead>
<tr>
<th>Encoding Modes and Data Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric code only</td>
</tr>
<tr>
<td>Alphanumeric</td>
</tr>
<tr>
<td>Binary (8 bits)</td>
</tr>
<tr>
<td>Kanji/Kana</td>
</tr>
</tbody>
</table>

VPE chooses automatically the best encoding. Only for Kanji you need to specify that the Kanji encoding shall be used.

**property QRCodeMode [integer] VPE.QRCodeMode**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

the encoding mode

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBAR2D_QRCODE_MODE_DEFAULT</td>
<td>2</td>
<td>Default</td>
<td>VPE chooses automatically between numeric, alphanumeric and binary mode</td>
</tr>
<tr>
<td>VBAR2D_QRCODE_MODE_KANJI</td>
<td>3</td>
<td>Kanji</td>
<td>Text is encoded in Kanji (Shift-JIS)</td>
</tr>
</tbody>
</table>

**Default:**

VBAR2D_QRCODE_MODE_DEFAULT

**See also:**

"Barcodes (2D)" in the Programmer's Manual
16.14 QRCodeBorder
[Professional Edition and above only]

Specifies the number of modules used for the border (usually 4) around QR Codes. The QR Code symbol area requires a border or "quiet zone" around it to be used. The border is a clear area around a symbol where nothing is printed. QR Code requires a four-module (or more) wide border at all sides of a symbol.

If you position no other objects near a QR Code, you might wish to set the border to zero, so the whole rectangle specified for the code is used for drawing the symbol pattern.

**property integer VPE.QRCodeBorder**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

the number of modules used for the border (usually 4)

**Default:**

4

**See also:**

"Barcodes (2D)" in the Programmer's Manual
16.15 QRCode

[Professional Edition and above only]

Inserts a QR Code barcode into the VPE document.

**method void VPE.QRCode(**

    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string Text

**)

*VpeCoord Left, Top, Right, Bottom*
coordinates

*string Text*
the text of the barcode

**Example:**

Doc.QRCode(1, 1, -3, -3, "Hello World")

**Remarks:**

sets **LastError**

**See also:**

"Barcodes (2D)" in the Programmer's Manual
16.16 RenderQRCode

[Professional Edition and above only]

Computes the dimensions of a QR Code barcode.

**method void VPE.RenderQRCode(**
  string Text,
  VpeCoord nWidth,
  VpeCoord nHeight,
  VpeCoord nModuleWidth

**)

**string Text**
the text of the barcode

**VpeCoord nWidth, nHeight**
With these two parameters you can specify a maximum size the barcode can have.

The following rules apply:

- nWidth and nHeight is specified: in this case the maximum rectangle of the barcode will be computed, which will fit into the given rectangle.
- Only nWidth is specified, nHeight is zero: In this case nHeight is computed (nWidth is also computed, i.e. adjusted).
- Only nHeight is specified, nWidth is zero: In this case nWidth is computed (nHeight is also computed, i.e. adjusted).
- nWidth and nHeight are zero: in this case the smallest possible rectangle is computed

**VpeCoord nModuleWidth**
If this parameter is zero, VPE will choose itself an optimum barcode module width. If it is non-zero, you can specify the width of a barcode module.

**Example:**

```csharp
// Compute the smallest possible rectangle for a given text
Doc.RenderQRCode("Hello World", 0, 0, 0)
xsize = Doc.nRenderWidth
ysize = Doc.nRenderHeight

// Insert the barcode into the document
Doc.QRCode(1, 1, -xsize, -ysize, "Hello World")
```

**Remarks:**
sets **LastError**

**See also:**
"Barcodes (2D)" in the Programmer's Manual
16.17 MaxiCode
[Professional Edition and above only]

Inserts a MaxiCode barcode into the VPE document.

This version accepts mode 2 and 3 messages beginning with “[])>S01G5”, which conform to particular open system standards (for details refer to AIM specification). It selects the appropriate mode based on the ZIP code included in Text.

**method void VPE.MaxiCode(**
  VpeCoord *Left*,
  VpeCoord *Top*,
  VpeCoord *Right*,
  VpeCoord *Bottom*,
  string *Text*
**)

*VpeCoord Left, Top, Right, Bottom*
coordinates

*string Text*
the text of the barcode

**Example:**

```cpp
// Sample Message (in C++ syntax):
char *szUPS = 
  "[])> "\x1e" "01" "\x1d" "9641460"
  "\x1d" "276" "\x1d" "068" "\x1d" "1Z51146547"
  "\x1d" "UPSN" "\x1d" "32630V" "\x1d" "327" "\x1d"
  "\x1d" "1/1" "\x1d" "2" "\x1d" "N" "\x1d"
  "\x1d" "NEUSS" "\x1d" "\x1e" "\x1d" "\x04"
  "!!!!!!!!!!!!!!!!!!!!!!"
Doc.MaxiCode(1, 6, -3, -3, szUPS)
```

**Remarks:**
sets **LastError**

**See also:**
"Barcodes (2D)" in the Programmer's Manual
### 16.18 RenderMaxiCode

[Professional Edition and above only]

Computes the dimensions of a MaxiCode barcode.

**method void VPE.RenderMaxiCode(**
  **string Text**
  **)**

**string Text**
  the text of the barcode

**Remarks:**

The width, height and module width of the MaxiCode barcode are fixed. You can not specify these parameters.

Sets **LastError**.

**Example:**

```c++
// Sample Message (in C++ syntax):
char *szUPS =
  "{}" "\xe0" "01" "\x1d" "9641460"
  "\x1d" "276" "\x1d" "068" "\x1d" "1251146547"
  "\x1d" "UPSN" "\x1d" "32630Y" "\x1d" "327" "\x1d"
  "\x1d" "1/1" "\x1d" "2" "\x1d" "N" "\x1d"
  "\x1d" "NEUSS" "\x1d" "\x1e" "\x04"
  "!!!!!!!!!!!!!!!!!!!!!!!!!!"

Doc.RenderMaxiCode(szUPS)
xsize = Doc.nRenderWidth
ysize = Doc.nRenderHeight
```

**See also:**

"Barcodes (2D)" in the Programmer's Manual
16.19 MaxiCodeEx

[Professional Edition and above only]

 Inserts a MaxiCode barcode into the VPE document. It is recommended to use MaxiCode() instead. This function has been implemented for completeness.

**method void VPE.MaxiCodeEx(**

VpeCoord `Left`,
VpeCoord `Top`,
VpeCoord `Right`,
VpeCoord `Bottom`,
int `nMode`,
string `Zip`,
int `nCountryCode`,
int `nServiceClass`,
string `Message`,
int `nSymbolNumber`,
int `nSymbolCount`

**)

**VpeCoord Left, Top, Right, Bottom coordinates**

**int nMode**

Modes define the structuring of data and error correction within a symbol.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Obsolete</td>
</tr>
<tr>
<td>1</td>
<td>Obsolete</td>
</tr>
<tr>
<td>2</td>
<td>Structured Carrier Message. Encode the destination address and class of service in the transport industry. Mode 2 supports up to nine numeric digits for the ZIP code.</td>
</tr>
<tr>
<td>3</td>
<td>Similar to mode 2. Supports up to six alphanumeric characters for ZIP code.</td>
</tr>
<tr>
<td>4</td>
<td>Standard Symbol (for details refer to AIM specification)</td>
</tr>
<tr>
<td>5</td>
<td>EEC (for details refer to AIM specification)</td>
</tr>
<tr>
<td>6</td>
<td>Reader Programming (for details refer to AIM specification)</td>
</tr>
</tbody>
</table>

**string Zip**

Postal code, for example D40211 or 42119

**int nCountryCode**

ISO 3166 country code (for example 276 for Germany)

**int nServiceClass**

Carrier-dependent service class

**string Message**

Secondary message

**int nSymbolNumber**

Total number of symbols (structured append)

**int nSymbolCount**

Current number of symbol (structured append)
Example:

```
Doc.MaxiCodeEx(1, 11, -5, -5, 2, "068107317", 840, 001, "AOE This is MaxiCode AOE", 1, 1)
```

Remarks:
sets `LastError`

See also:
"Barcodes (2D)" in the Programmer's Manual
16.20 RenderMaxiCodeEx

[Professional Edition and above only]

Computes the dimensions of a MaxiCode barcode.

**method void VPE.MaxiCodeEx(**
int nMode,
string Zip,
int nCountryCode,
int nServiceClass,
string Message,
int nSymbolNumber,
in nSymbolCount
)**

**int nMode**

Modes define the structuring of data and error correction within a symbol.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Obsolete</td>
</tr>
<tr>
<td>1</td>
<td>Obsolete</td>
</tr>
<tr>
<td>2</td>
<td>Structured Carrier Message. Encode the destination address and class of service in the transport industry. Mode 2 supports up to nine numeric digits for the ZIP code.</td>
</tr>
<tr>
<td>3</td>
<td>Similar to mode 2. Supports up to six alphanumeric characters for ZIP code.</td>
</tr>
<tr>
<td>4</td>
<td>Standard Symbol (for details refer to AIM specification)</td>
</tr>
<tr>
<td>5</td>
<td>EEC (for details refer to AIM specification)</td>
</tr>
<tr>
<td>6</td>
<td>Reader Programming (for details refer to AIM specification)</td>
</tr>
</tbody>
</table>

**string Zip**
Postal code, for example D40211 or 42119

**int nCountryCode**
ISO 3166 country code (for example 276 for Germany)

**int nServiceClass**
Carrier-dependent service class

**string Message**
Secondary message

**int nSymbolNumber**
Total number of symbols (structured append)

**int nSymbolCount**
Current number of symbol (structured append)

**Remarks:**
The width, height and module width of the MaxiCode barcode are fixed. You can not specify these parameters.

Sets **LastError**, **LastError**.
Example:

```vbnet
Doc.RenderMaxiCodeEx(1, 11, -5, -5, 2, "068107317", 840, 001,
                      "AOE This is MaxiCode AOE", 1, 1)
xsize = Doc.nRenderWidth
ysize = Doc.nRenderHeight
```

See also:

"Barcodes (2D)" in the Programmer's Manual
16.21 PDF417ErrorLevel

[Professional Edition and above only]
Sets the error level for PDF417 barcodes.

property integer VPE.PDF417ErrorLevel

    read / write; runtime only; also supported by TVPEObject

Possible Values:
The higher the level, the more redundancy is added to the symbol allowing more errors to be corrected at the expense of less user data capacity. If the level is -1, an error level is chosen automatically based on the message length as recommended by the AIM specification.

Default:
-1

See also:
"Barcodes (2D)" in the Programmer's Manual
16.22 PDF417Rows

[Professional Edition and above only]
Sets the desired number of matrix rows (0: choose automatically based on message).

property integer VPE.PDF417

     read / write; runtime only; also supported by TVPEObject

Possible Values:
     Desired number of matrix rows (0: choose automatically based on message).

Default:
     0

See also:
     "Barcodes (2D)" in the Programmer's Manual
16.23 PDF417Columns

[Professional Edition and above only]
Sets the desired number of matrix columns (0: choose automatically based on message).

**property integer VPE.PDF417Columns**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**
Desired number of matrix columns (0: choose automatically based on message).

**Default:**
0

**See also:**
"Barcodes (2D)" in the Programmer's Manual
16.24 PDF417  
[Professional Edition and above only]  
Inserts a PDF417 barcode into the VPE document.

**method void VPE.PDF417(**
  VpeCoord Left,
  VpeCoord Top,
  VpeCoord Right,
  VpeCoord Bottom,
  string Text
)

*VpeCoord Left, Top, Right, Bottom*  
coordinates

*string Text*  
the text of the barcode

**Example:**

```
szStr =
"01\t05\t()-\t82\tWSP3.5.1DE\r"
"02\t23112001\t1232630V6851146547\t08\tM\t9838571153\t3\t1.5\tKgs\t1.5\tP/P\t0.00\t\tDE\t\t\t\r"
"04\tSH\tX-LOGISTIK GMBH\tWALLENHORST\t49134\tDE\t32630V\tC/O INTERTRADE AG\t+495407-834343\t\tAM OHLENBERG 14\t\r"
"04\tST\tIDEAL SOFTWARE GMBH\tNEUSS\t41464\tDE\t32640\tMR. XXX\t\t\r"
"99\r";
Doc.PDF417(1, 17, -5.5, -3.5, szStr);
```

**Remarks:**  
sets **LastError**

**See also:**  
"Barcodes (2D)" in the Programmer's Manual
16.25 RenderPDF417

[Professional Edition and above only]

Computes the dimensions of a PDF417 barcode.

**method void VPE.RenderPDF417(**

  **string Text,**
  **VpeCoord nWidth,**
  **VpeCoord nHeight,**
  **VpeCoord nModuleWidth**

  **)**

  **string Text**
  
  the text of the barcode

  **VpeCoord nWidth, nHeight**
  
  With these two parameters you can specify a maximum size the barcode can have.

  The following rules apply:

  - nWidth and nHeight is specified: in this case the maximum rectangle of the barcode will be computed, which will fit into the given rectangle.
  - Only nWidth is specified, nHeight is zero: In this case nHeight is computed (nWidth is also computed, i.e. adjusted).
  - Only nHeight is specified, nWidth is zero: In this case nWidth is computed (nHeight is also computed, i.e. adjusted).
  - nWidth and nHeight are zero: in this case the smallest possible rectangle is computed

  **VpeCoord nModuleWidth**
  
  If this parameter is zero, VPE will choose itself an optimum barcode module width. If it is non-zero, you can specify the width of a barcode module.

**Example:**

```csharp
szStr = "01\t05\t\t82\tWSP3.5.1DE\r"
  "02\t23112001\t123732630V6851146547\t08\tM\t9838571513\t3\t1.5\tKgs\t1.5\tP/P\t0.00\t\t\t\t\t\t\t\t\t\t\t\t"
  "04\tS\tX-LOGISTIK GMBH\tWALLENHORST\t9134\tDE\t32630V\tC/O INTERTRADE AG\t\t\t\tHERR ERNST LEMKE\t+495407-834343\tAM OHLENBERG 14\t"n"
  "04\tS\tT\tIDEAL SOFTWARE GMBH\tNEUSS\t41464\tDE\t102A\tMR. XXX\tMR. YYYY\t"
  "99\r";

// Compute the smallest possible rectangle for a given
textDoc.RenderPDF417(szStr, 0, 0, 0)
xsize = Doc.nRenderWidth
ysize = Doc.nRenderHeight

// Insert the barcode into the document
Doc.PDF417(1, 1, -xsize, -ysize, szStr)
```

**Remarks:**

- sets **LastError**
See also:
"Barcodes (2D)" in the Programmer's Manual
16.26 AztecFlags

[Professional Edition and above only]

Sets the flag for the interpretation of the Text string parameter of the `Aztec()}-method.

**property boolean VPE.AztecFlags**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

If true, the lpszString parameter of the `Aztec()`-function uses "<Esc>n" for FLG(n), and "<Esc><Esc>" for "<Esc>"

**Default:**

False

**See also:**

"Barcodes (2D)" in the Programmer's Manual
16.27 AztecControl

[Professional Edition and above only]
Sets the flag for the interpretation of the lpszString parameter of the <code>Aztec()</code>-function.

**property integer VPE.AztecControl**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>default error correction level</td>
</tr>
<tr>
<td>01 to 99</td>
<td>minimum error correction percentage</td>
</tr>
<tr>
<td>101 to 104</td>
<td>1 to 4-layer Compact symbol</td>
</tr>
<tr>
<td>201 to 232</td>
<td>1 to 32-layer Full-Range symbol</td>
</tr>
<tr>
<td>300</td>
<td>a simple Aztec &quot;Rune&quot;</td>
</tr>
</tbody>
</table>

**Default:**

0

**See also:**

"Barcodes (2D)" in the Programmer's Manual
16.28 AztecMenu

[Professional Edition and above only]

Specifies, if this is a "Menu" symbol.

**property boolean VPE.AztecMenu**

   read / write; runtime only; also supported by TVPEObject

**Possible Values:**

   specifies, if this is a "Menu" symbol

**Default:**

   False

**See also:**

   "Barcodes (2D)" in the Programmer's Manual
16.29 AztecMultipleSymbols

[Professional Edition and above only]

Specifies # of symbols for message… 1 (normal) to 26.

**property integer VPE.AztecMultipleSymbols**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**

# of symbols for message… 1 (normal) to 26

**Default:**

1

**See also:**

"Barcodes (2D)" in the Programmer's Manual
16.30 AztecID
[Professional Edition and above only]
Specifies an optional null-terminated ID field for append.

property string VPE.AztecID

write; runtime only; also supported by TVPEObject

Possible Values:
optional ID field for append

Default:
NULL (empty)

See also:
"Barcodes (2D)" in the Programmer's Manual
16.31 Aztec

[Professional Edition and above only]

Inserts an Aztec barcode into the VPE document.

```java
method void VPE.Aztec(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string Text
)
```

*VpeCoord Left, Top, Right, Bottom*
coordinates

*string Text*
the text of the barcode

**Example:**

```
Doc.Aztec(1, 23, -3, -3, "This is a message")
```

**Remarks:**

sets `LastError`

**See also:**

"Barcodes (2D)" in the Programmer's Manual
16.32 RenderAztec

[Professional Edition and above only]

Computes the dimensions of an **Aztec** barcode.

**method void VPE.RenderAztec**(  
  string *Text*,  
  VpeCoord *nWidth*,  
  VpeCoord *nHeight*,  
  VpeCoord *nModuleWidth*  
)

*string Text*  
the text of the barcode

*VpeCoord nWidth, nHeight*  
With these two parameters you can specify a maximum size the barcode can have.

The following rules apply:

- nWidth and nHeight is specified: in this case the maximum rectangle of the barcode will be computed, which will fit into the given rectangle.
- Only nWidth is specified, nHeight is zero: In this case nHeight is computed (nWidth is also computed, i.e. adjusted).
- Only nHeight is specified, nWidth is zero: In this case nWidth is computed (nHeight is also computed, i.e. adjusted).
- nWidth and nHeight are zero: in this case the smallest possible rectangle is computed

*VpeCoord nModuleWidth*  
If this parameter is zero, VPE will choose itself an optimum barcode module width. If it is non-zero, you can specify the width of a barcode module.

**Example:**

```
// Compute the smallest possible rectangle for a given text
Doc.RenderAztec("This is a message", 0, 0, 0)
xsize = Doc.nRenderWidth
ysize = Doc.nRenderHeight

// Insert the barcode into the document
Doc.Aztec(1, 1, -xsize, -ysize, "This is a message")
```

**Remarks:**

sets *LastError*

**See also:**

"Barcodes (2D)" in the Programmer's Manual
17 E-Mail Functions

17.1 E-Mail Functions

[Windows platform only; not supported by the Community Edition]

VPE allows to send e-mails with VPE Documents via MAPI. Additionally, the e-mail feature allows very easily to fax VPE Documents directly with MS-FAX and MS-MAIL through MAPI.

E-mailing and faxing can be done with user interaction (mail dialog is shown) or without user interaction - only by code. You can specify by code:

- the sender
- a list of receivers (also CC and BCC)
- the message subject
- the message text
- a list of attachments (including or excluding VPE Documents)

Even if mailing is done with user interaction (mail dialog is shown), the properties you had set by code are valid and will be shown in the mail dialog.

VPE includes VPE View (see "VPE View - the Document Viewer" in the Programmer's Manual) a royalty-free Document Viewer for VPE Documents, which is very useful if documents are sent by e-mail, so the recipient can view the documents with a single mouse click. VPE View is also required if VPE Documents are faxed by using VPE's e-mail functions.

Note:

VPE is not an e-mail component. It does only provide basic e-mail support. If you need extended e-mailing features, process the event VPE_BEFORE_MAIL and cancel the event. After cancelling the event you can execute your own mailing code, using sophisticated MAPI components. To attach for example the current VPE document as PDF to an e-mail created by a MAPI component, simply write the current VPE document to a temporary file using WriteDoc() and attach it to the e-mail. You could even generate and attach at this moment a new temporary VPE document with different content, for example with annotations or a watermark.
17.2 Sending Mail on 64-Bit Windows

VPE is using the Simple MAPI or Extended MAPI subsystem of the Windows operating system. Due to the design of both application interfaces, either subsystem is bound to the CPU architecture, for which your application is compiled / executed.

At the time of this writing, this means: the 64-bit version of your applications require a 64-bit mail client, and the 32-bit version of your applications require a 32-bit mail client.

Maybe Microsoft - or the developers of mail clients - will address this issue in a later version and provide bridging from one CPU architecture to the other.
17.3 IsMAPIInstalled

[Windows platform only; not supported by the Community Edition]

Checks the machine if MAPI is installed. This property is obsolete, use the property MAPIType instead.

**property MAPIStatus [integer] VPE.IsMAPIInstalled**

read; runtime only

**Returns:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMAPI_NOT_INSTALLED</td>
<td>0</td>
<td>NotInstalled</td>
<td>MAPI is not installed</td>
</tr>
<tr>
<td>VMAPI_INSTALLED</td>
<td>1</td>
<td>Installed</td>
<td>MAPI is installed</td>
</tr>
<tr>
<td>VMAPI_UNSURE</td>
<td>2</td>
<td>Unsure</td>
<td>unsure (obsolete, was for 16-bit version)</td>
</tr>
</tbody>
</table>

**Remarks:**

In case of an error, LastError is set.

There are some problems with MAPI. Some vendors of MAPI clients (e-mail software) do not set up the registry correctly. The problems are sometimes very heavy and difficult to solve. In the following we list the problems we heared about:

**32-bit VPE on Win-32 (Win9x / ME / NT >= 4.0 / 2000 / XP) platform**

MAPI clients are not correctly registered. The guideline is, that the following entry must be present in the registry for all 32-bit platforms listed above:

HKEY_LOCAL_MACHINE \ SOFTWARE \ Microsoft \ Windows Messaging Subsystem \ MAPI = 1

But for NT <= v4.00 there has to be the following entry in the WIN.INI in the section [Mail]:

[Mail]
MAPI = 1

VPE checks for all conditions above, but some MAPI clients do not setup the registry as they should.

To cure the problem listed above, we implemented the following solution:

VPE checks for the guideline rules first and returns VMAPI_INSTALLED if they are matched. Otherwise VPE will check for a WIN.INI entry (yes!) which is made by some clients in the [Mail] section:

[Mail]
MAPIX = 1

If this entry is found, VPE will return VMAPI_UNSURE, because it is not 100% sure that a 32-bit MAPI client is really installed. Otherwise VPE will return VMAPI_NOT_INSTALLED. If you are sure there is a working MAPI client on the machine, but the registry key is missing, you can add it manually or let this perform by your setup software.

The e-mail toolbar button and the method MailDoc()
The e-mail button will be enabled for the condition VMAPI_INSTALLED and will be disabled (grayed) for the conditions VMAPI_UNSURE and VMAPI_NOT_INSTALLED.

You have the chance to re-enable the e-mail button after calling OpenDoc(), if you are sure a working MAPI client is installed (see EnableMailButton).

The method MailDoc() will work for all conditions (i.e. it is not blocked), but for VMAPI_NOT_INSTALLED and VMAPI_UNSURE you call it on your own risc.
17.4 MAPIType

[Windows platform only; not supported by the Community Edition]

Checks the machine, what type of MAPI is installed. This property supersedes the property IsMAPIInstalled and should be used instead.

property MAPIType [integer] VPE.MAPIType

   read / write; runtime only

Returns:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMAPI_TYPE_NOT_INSTALLED</td>
<td>0</td>
<td>NotInstalled</td>
<td>MAPI is not installed</td>
</tr>
<tr>
<td>VMAPI_TYPE_EXTENDED</td>
<td>1</td>
<td>Extended</td>
<td>Extended MAPI is used</td>
</tr>
<tr>
<td>VMAPI_TYPE_SIMPLE</td>
<td>2</td>
<td>Simple</td>
<td>Simple MAPI is used</td>
</tr>
</tbody>
</table>

Remarks:

Extended MAPI and Simple MAPI are both APIs designed by Microsoft. Extended MAPI is a rather complex API, which is not supported by some mail clients, e.g. Thunderbird. VPE determines the MAPI type which is supported by the default mail client, and returns it in this property. Sending the body of an e-mail message (this is different from attaching a VPE document as exported HTML) as HTML or RTF is only supported reliably, if Extended MAPI is supported.

If both MAPI types are available on the machine, VPE selects Simple MAPI by default. Due to the rather chaotic infrastructure of the Windows messaging subsystem, Simple MAPI causes less problems.
17.5 MailSender

[Windows platform only; not supported by the Community Edition]
Allows you to specify the message sender by code.

property string VPE.MailSender

    write; runtime only

Possible Values:
the name / e-mail address of the sender

Remarks:
Normally it is not necessary to set this property, because the MAPI client will use the default profile, or the user will be able to select his profile.

If Extended MAPI is active, setting this property has no effect.

Example:

```
Doc.MailSender = "Support@IdealSoftware.com"
```
17.6 AddMailReceiver

[Windows platform only; not supported by the Community Edition]

Allows you to specify a list of e-mail receivers by code.

method void VPE.AddMailReceiver(
    string Receiver,
    RecipientClass [long] RecipientClass
)

string Receiver
    the name / e-mail address of the receiver

RecipientClass [long] RecipientClass
    possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMAIL_ORIG</td>
<td>0</td>
<td>Orig</td>
<td>Recipient is message originator</td>
</tr>
<tr>
<td>VMAIL_TO</td>
<td>1</td>
<td>To</td>
<td>Recipient is a primary recipient</td>
</tr>
<tr>
<td>VMAIL_CC</td>
<td>2</td>
<td>Cc</td>
<td>Recipient is a copy recipient</td>
</tr>
<tr>
<td>VMAIL_BCC</td>
<td>3</td>
<td>Bcc</td>
<td>Recipient is blind copy recipient</td>
</tr>
<tr>
<td>VMAIL_RESOLVE_NAME</td>
<td>-2147483648</td>
<td>ResolveName</td>
<td>try to resolve the name from the mail client's address book (Simple MAPI only)</td>
</tr>
</tbody>
</table>

Remarks:
The list of recipients will be valid until ClearMailReceivers() is called.

Simple MAPI:
Due to the security features in Outlook 2002 or higher (which can also be installed for earlier Outlook versions as security fix), Outlook will show a warning message that an external application is trying to access the address book, when sending an e-mail. To avoid this, VPE does not call MAPIResolveName() anymore. If you want VPE to look up a name in the mail client's address book, add VMAIL_RESOLVE_NAME to the RecipientClass parameter.

E.g. Doc.AddMailReceiver("IDEAL Software", VMAIL_TO + VMAIL_RESOLVE_NAME)

Will resolve the receiver to "Support@IdealSoftware.com" if "IDEAL Software" is stored in the address book of the MAPI client as "Support@IdealSoftware.com".

However, if you use this flag - with Outlook 2002 or higher or the security fix installed - this will cause a warning message to appear that an external application is trying to access the address book.

The warning is also shown by Outlook - and can not be avoided - if you sent e-mails without showing a dialog (i.e. MailWithDialog = False).

The warning can be disabled in Outlook with "Options | Security" by unchecking "Show warning if other applications try to send an e-mail under my name".

If you are not using VMAIL_RESOLVE_NAME and you are using Outlook / Exchange, make sure SMTP is properly configured, otherwise an error message might appear that the message could not be delivered.

Please note that VMAIL_RESOLVE_NAME is not supported, if VPE uses Extended MAPI.
Example:

**ActiveX / VCL:**

```c
Doc.AddMailReceiver("max@alpha.com", VMAIL_CC)
```

**.NET:**

```c
Doc.AddMailReceiver("max@alpha.com", RecipientClass.Cc)
```

Will add "max@alpha.com" as carbon copy recipient for the same e-mail.

**ActiveX / VCL:**

```c
Doc.AddMailReceiver("[FAX: +49 1234 12345678]", VMAIL_TO)
```

**.NET:**

```c
Doc.AddMailReceiver("[FAX: +49 1234 12345678]", RecipientClass.To)
```

Will **FAX** the message to the given phone number! If a VPE Document is attached to the mail, the VISIBLE CONTENT of the document will be faxed. Requires MS-MAIL and MS-FAX, or a similar MAPI- Mail / Fax software to be installed on the system. Also the VPE Document viewer VPE View (see "VPE View - the Document Viewer" in the Programmer's Manual) needs to be correctly installed, otherwise VPE Document attachments will not be faxed with their visual content.
17.7 ClearMailReceivers

[Windows platform only; not supported by the Community Edition]

Clears the list of mail recipients created with AddMailReceiver().

```java
method void VPE.ClearMailReceivers()
```

17.8 AddMailAttachment

[Windows platform only; not supported by the Community Edition]

Allows you to specify a list of e-mail attachments (files) by code.

**method void VPE.AddMailAttachment(**

    String PathName,
    String FileName

)**

*String PathName*

The fully qualified path and file name of the attachment file. This path should include the disk drive letter and directory name. The attachment file should be closed before this call is made.

When **MailDoc()** is called, VPE will examine all attachment paths. If the last character of a path is a backslash ("\"), VPE will write the current document as a temporary file to the specified path and add it as attachment to the mail.

If the examined path is NULL or empty (""), VPE will create and attach the temporary document in the temporary directory specified by the environment variable named TMP or TEMP, or - if both are not set - in the current working directory.

Any temporarily created file will be deleted before the MailDoc() method returns.

Otherwise - if a fully qualified path and file name was specified - VPE expects that the attachment file has already been created. This gives you the possibility to mail any kind of attachment, not being limited to VPE-Documents.

*String FileName*

The filename seen by the recipient. This name can differ from the filename in PathName if temporary files are being used. If FileName is NULL or empty (""), the filename from PathName is used. If the attachment is an OLE object, FileName contains the class name of the object, such as "Microsoft Excel Worksheet."

**Remarks:**

By default, the current document is automatically set as an attachment located in the tmp-dir (the file will only be created if - and in the moment when - MailDoc() is executed). The file name of the attachment is either the Application Name or - if **SwapFileName** is set - the document file name. To clear that default, call **ClearMailAttachments()** before specifying your own attachments.

The property **MailAutoAttachDocType** controls, whether a VPE Document or a PDF Document is attached to the e-mail.

**Examples:**

```csharp
Doc.ClearMailAttachments()
```

Deletes the default attachment, which is the VPE Document itself.

```csharp
Doc.AddMailAttachment("c:\important_file.zip", "")
```

Sets "c:\important_file.zip" as the ONLY attachment. The VPE Document is not sent, because ClearMailAttachments() deleted the default attachment of the VPE Document.

```csharp
Doc.AddMailAttachment("c:\important_file.txt", "")
```

Adds "c:\important_file.txt" as second attachment.
Doc.WriteDoc("c:\tmp\doc.vpe")
Doc.AddMailAttachment("c:\tmp\doc.vpe", ")

 Adds "c:\tmp\doc.vpe" as third attachment.
17.9 MailAutoAttachDocType

[Windows platform only; not supported by the Community Edition]

The value of this property controls, what type of document is automatically attached to e-mails
(see also AddMailAttachment).

**property DocExportType [integer] VPE.MailAutoAttachDocType**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPE_DOC_TYPE_AUTO</td>
<td>0</td>
<td>Auto</td>
<td>MailDoc() will attach the document as VPE, PDF or HTML file, depending on the file suffix of the parameter file_name which is supplied to AddMailAttachment().</td>
</tr>
<tr>
<td>VPE_DOC_TYPE_VPE</td>
<td>1</td>
<td>VPE</td>
<td>MailDoc() will attach the document as VPE document file, regardless of the file suffix</td>
</tr>
<tr>
<td>VPE_DOC_TYPE_PDF</td>
<td>2</td>
<td>PDF</td>
<td>MailDoc() will attach the document as PDF document file, regardless of the file suffix</td>
</tr>
<tr>
<td>VPE_DOC_TYPE_HTML</td>
<td>3</td>
<td>HTML</td>
<td>MailDoc() will attach the document as HTML document file, regardless of the file suffix</td>
</tr>
<tr>
<td>VPE_DOC_TYPE_XML</td>
<td>4</td>
<td>XML</td>
<td>VPE XML Format</td>
</tr>
<tr>
<td>VPE_DOC_TYPE_ODT</td>
<td>5</td>
<td>ODT</td>
<td>OpenOffice Document Text</td>
</tr>
</tbody>
</table>

**Default:**

VPE_DOC_TYPE_PDF
17.10 ClearMailAttachments

[Windows platform only; not supported by the Community Edition]

Clears the list of mail attachments created with `AddMailAttachment`.

```java
method void ClearMailAttachments()
```

17.11 MailSubject

[Windows platform only; not supported by the Community Edition]

Allows you to specify the e-mail subject by code.

property string VPE.MailSubject

write; runtime only

Possible Values:
the subject of the message

Example:

```csharp
Doc.MailSubject = "New VPE v3.0"
```
17.12 MailText
[Windows platform only; not supported by the Community Edition]

Allows you to specify the e-mail text by code.

**property string VPE.MailText**

write; runtime only

**Possible Values:**
the message text

**Remarks:**
If VPE is using Extended MAPI (see [MAPIType](#)), you can also use **HTML** or **RTF** for the body text of e-mail messages. For HTML the message text must begin with "<html>" and for RTF the message text must begin with "\rtf".

The Community Edition does not support Extended MAPI.

**Example:**

```vba
Doc.MailText = "Dear customer of VPE," + chr$(13) + chr$(10) + "we have a new version released."
```
17.13 MailWithDialog

[Windows platform only; not supported by the Community Edition]

Specifies, whether a dialog is shown to the user with input fields for recipients, subject, text and attachments when an e-mail is sent. An e-mail is sent either by pushing the e-Mail button in the preview or by executing MailDoc() by code. It is very useful to set MailWithDialog to false, if you do automatic serial e-mailings in the background without user interaction.

property boolean VPE.MailWithDialog

write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>show dialog</td>
</tr>
<tr>
<td>False</td>
<td>do not show the dialog</td>
</tr>
</tbody>
</table>

Default:

True

Remarks:

If the user has not logged on to a MAPI client, a logon dialog might be displayed once for the first e-mail that is sent, regardless of the setting of this property.

If you are setting MailWithDialog = False, at least one recipient must be specified by code (see AddMailReceiver()), otherwise a dialog is shown.

This option does not work with some MAPI clients (a dialog will always be shown). It also depends sometimes on the MAPI client configuration, whether a dialog is shown or not.
17.14 MailDoc

[Windows platform only; not supported by the Community Edition]

Sends a mail with the current Mail-Properties.

```
method boolean VPE.MailDoc(
  )
```

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success</td>
</tr>
<tr>
<td>False</td>
<td>failure</td>
</tr>
</tbody>
</table>

**Remarks:**

In case of an error, LastError is set.

The name of the default attachment is either the DevJobName (if it is set), or the title of the VPE preview window.

There are some problems with MAPI clients (e-mail software). This may cause that MailDoc() will return an error and / or that the e-mail button in the toolbar is disabled (grayed).

Recipients, subject, text and attachments specified by code will be shown in the MAPI client dialog (for example Microsoft Exchange or Outlook).

Not all MAPI clients return correct error codes. Some of them return "success", even if the user aborted. Others return "common failure" if an attached file is not existing, instead of "attachment not found", etc.

Netscape Messenger and the Mozilla mail client seem not to work correctly as MAPI clients.

*MAPI clients are manufactured by vendors independent of IDEAL Software; we make no warranty, implied or otherwise, regarding these product's performance or reliability.*

Special care must be taken, when creating VPE Document files that contain pictures or UDO's. For details please see WriteDoc.

**Example:**

```c
Doc.MailSubject = "VPE Demo"
Doc.AddMailReceiver("MrX@dummyxyz.com", VMAIL_TO)
Doc.AddMailReceiver("MrY@dummyxyz.com", VMAIL_CC)
Doc.AddMailAttachment("c:\data\report.vpe", ")
Doc.MailDoc()
```

```c
Doc.AddMailReceiver("[FAX: +49 1234 12345678]", VMAIL_TO)
Doc.MailDoc()
```

Will FAX the message to the given phone number! If a VPE Document is attached to the mail, the VISIBLE CONTENT of the document will be faxed. Requires MS-MAIL and MS-FAX, or a similar MAPI Mail / Fax software to be installed on the system. Also the VPE Document viewer VPE View see "VPE View - the Document Viewer" in the Programmer's Manual) needs...
to be correctly installed, otherwise VPE Document attachments will not be faxed with their visual content.
18 RTF Functions

18.1 RTF Functions
[Professional Edition and above]

Features of VPE
- Version independent and error tolerant RTF parser. This means: no termination, if unknown or erroneous formats are processed, instead they are ignored and skipped until the next known keyword is found.
- Any font-types, -sizes and -attributes (bold, italic underlined) can be used even in a single line or word
- Text- and background color can be set for each letter
- Unlimited number of paragraphs (only limited by available memory)
- Automatic text break over multiple pages

For an introduction into RTF and a detailed description of VPE's RTF features, see "RTF - Rich Text Format" in the Programmer's Manual

NOTE: Often RTF documents are created on Windows platforms and use Windows specific TrueType fonts. Especially for the use on Non-Windows platforms there is the important method SetFontSubstitution() available to substitute fonts, so you can use other fonts in place of the fonts used within an RTF document.
18.2 WriteRTF

Outputs RTF text within a rectangle at position Left, Top with the right border at Right and the bottom border at Bottom.

The pen is invisible.

method VpeCoord VPE.WriteRTF(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string Text
)

VpeCoord Left, Top, Right, Bottom
position and dimensions

string Text
the string to output

Returns:
the bottom y-coordinate generated by the output

Remarks:
VFREE: only the Bottom coordinate may be set to VFREE, not the Right coordinate.
In case of an error, LastError is set.

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

Example:

ActiveX / VCL:

```
Doc.SetFont("Arial", 12)
Doc.WriteRTF(1, 1, -5, VFREE, "Hello \b World!")
Doc.SetFont("Times New Roman", 16)
Doc.TextUnderline = True
Doc.WriteRTF(VLEFT, VBOTTOM, VRIGHT, VFREE, "Hello \b World!")
```

.NET:

```
Doc.SetFont("Arial", 12)
Doc.WriteRTF(1, 1, -5, Doc.nFree, "Hello \b World!")
Doc.SetFont("Times New Roman", 16)
Doc.TextUnderline = True
Doc.WriteRTF(Doc.nLeft, Doc.nBottom, Doc.nRight, Doc.nFree, "Hello \b World!")
```

Produces the following output:

Hello World!
Hello World!
See also:
18.3 WriteBoxRTF

Same as WriteRTF(), but pen- and box-settings are used. Outputs RTF text within a rectangle at position Left, Top with the right border at Right and the bottom border at Bottom.

method VpeCoord VPE.WriteBoxRTF(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string Text
)

VpeCoord Left, Top, Right, Bottom
    position and dimensions

string Text
    the string to output

Returns:
    the bottom y-coordinate generated by the output

Remarks:
    VFREE: only the Bottom coordinate may be set to VFREE, not the Right coordinate.
    In case of an error, LastError is set.

    VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

See also:
18.4 WriteRTFFile

Outputs RTF text, which is read from the file specified in FileName, within a rectangle at position Left, Top with the right border at Right and the bottom border at Bottom. This is very useful for processing text already created by your end user with an RTF editor.

The pen is invisible.

```vpe
method VpeCoord VPE.WriteRTFFile(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string FileName
)
```

| VpeCoord Left, Top, Right, Bottom | position and dimensions |
| string FileName                  | the file with RTF text that is imported |

**Returns:**
the bottom y-coordinate generated by the output

**Remarks:**
VFREE: only the Bottom coordinate may be set to VFREE, not the Right coordinate.

Keywords in the RTF file override any VPE API properties. For example, if you set "f1 = Arial" and in the RTF file it is defined as "f1 = Times New Roman", then "f1" will be "Times New Roman".

In case of an error, `LastError` is set.

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

**See also:**
18.5 WriteBoxRTFFile

Same as WriteRTFFile(), but pen- and box-settings are used. Outputs RTF text, which is read from the file specified in FileName, within a rectangle at position Left, Top with the right border at Right and the bottom border at Bottom. This is very useful for processing text already created by your end user with an RTF editor.

method VpeCoord VPE.WriteBoxRTFFile(
   VpeCoord Left,
   VpeCoord Top,
   VpeCoord Right,
   VpeCoord Bottom,
   string FileName
)

VpeCoord Left, Top, Right, Bottom
position and dimensions

string FileName
the file with RTF text that is imported

Returns:
VFREE: only the Bottom coordinate may be set to VFREE, not the Right coordinate.
the bottom y-coordinate generated by the output

Remarks:
Keywords in the RTF file override any VPE API properties. For example, if you set "\f1 = Arial" and in the RTF file it is defined as "\f1 = Times New Roman", then "\f1" will be "Times New Roman".

In case of an error, LastError is set.

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

See also:
18.6 WriteRTFStream

Outputs RTF text, which is read from the supplied stream, within a rectangle at position Left, Top with the right border at Right and the bottom border at Bottom. This is very useful for processing rich text stored in a database.

The pen is invisible.

```
method VpeCoord VPE.WriteRTFStream(
    TVPEStream Stream,
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom
)
```

TVPEStream Stream
stream object

VpeCoord Left, Top, Right, Bottom
position and dimensions

Returns:
the bottom y-coordinate generated by the output

Remarks:
VFREE: only the Bottom coordinate may be set to VFREE, not the Right coordinate.

Keywords in the RTF stream override any VPE API properties. For example, if you set "\f1 = Arial" and in the RTF stream it is defined as "\f1 = Times New Roman", then "\f1" will be "Times New Roman".

In case of an error, LastError is set.

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

See also:
18.7 WriteBoxRTFStream

Same as WriteRTFStream(), but pen- and box-settings are used. Outputs RTF text, which is read from the supplied stream, within a rectangle at position Left, Top with the right border at Right and the bottom border at Bottom. This is very useful for processing rich text stored in a database.

```vpe
def WriteBoxRTFStream(stream, left, top, right, bottom):
    return
```

**TVPEStream Stream**

stream object

**VpeCoord Left, Top, Right, Bottom**

position and dimensions

**Returns:**

VFREE: only the Bottom coordinate may be set to VFREE, not the Right coordinate.

the bottom y-coordinate generated by the output

**Remarks:**

Keywords in the RTF stream override any VPE API properties. For example, if you set "\f1 = Arial" and in the RTF stream it is defined as "\f1 = Times New Roman", then "f1" will be "Times New Roman".

In case of an error, LastError is set.

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content.

For details please see "Dynamic Positioning" in the Programmer's Manual.

**See also:**

18.8 SetRTFFont
Modify the build-in RTF font table.

method void VPE.SetRTFFont(
    long ID,
    string Name
)

long ID
    the entry ID
string Name
    the name of the font that shall be assigned to the given entry-id

Remarks:
The current value of the CharSet property is also stored in the font table entry when calling this method.

VPE has build-in a predefined font table:
\f1 is predefined as "Arial"
\f2 is predefined as "Times New Roman"

Examples:
For example change \f1 with:

ActiveX / VCL:
Doc.charset = SYMBOL_CHARSET
Doc.SetRTFFont(1, "Wingdings")

.NET:
Doc.charset = CharSet.Symbol
Doc.SetRTFFont(1, "Wingdings")

Or you can add as many fonts to the table as you like, for example:

ActiveX / VCL:
Doc.charset = SYMBOL_CHARSET
Doc.SetRTFFont(3, "Wingdings")

.NET:
Doc.charset = CharSet.Symbol
Doc.SetRTFFont(1, "Wingdings")

See also:
18.9 SetRTFColor

Modify the build-in RTF color table.

```
method void VPE.SetRTFColor(
    long ID,
    Color Color
)
```

*long ID*
the entry ID

*Color Color*
any of the "COLOR_xyz" constants described in Programmer's Manual
or ActiveX / VCL: any RGB value
or .NET: any member of the .NET Color structure
that shall be assigned to the given entry-id

Remarks:
RTF offers no way to set a transparent background mode, like VPE does. So we decided to
set the background mode to transparent, in case you are using COLOR_WHITE = RGB(255,
255, 255) as background color.

VPE has build-in a predefined color table:

<table>
<thead>
<tr>
<th>Color</th>
<th>RGB</th>
</tr>
</thead>
<tbody>
<tr>
<td>color1</td>
<td>BLACK</td>
</tr>
<tr>
<td>color2</td>
<td>DKGRAY</td>
</tr>
<tr>
<td>color3</td>
<td>GRAY</td>
</tr>
<tr>
<td>color4</td>
<td>LTGRAY</td>
</tr>
<tr>
<td>color5</td>
<td>WHITE</td>
</tr>
<tr>
<td>color6</td>
<td>DKRED</td>
</tr>
<tr>
<td>color7</td>
<td>RED</td>
</tr>
<tr>
<td>color8</td>
<td>LTRED</td>
</tr>
<tr>
<td>color9</td>
<td>DKORANGE</td>
</tr>
<tr>
<td>color10</td>
<td>ORANGE</td>
</tr>
<tr>
<td>color11</td>
<td>LTORANGE</td>
</tr>
<tr>
<td>color12</td>
<td>DKYELLOW</td>
</tr>
<tr>
<td>color13</td>
<td>YELLOW</td>
</tr>
<tr>
<td>color14</td>
<td>LTYELLOW</td>
</tr>
<tr>
<td>color15</td>
<td>DKGREEN</td>
</tr>
<tr>
<td>color16</td>
<td>GREEN</td>
</tr>
<tr>
<td>color17</td>
<td>LTGREEN</td>
</tr>
<tr>
<td>color18</td>
<td>HIGREEN</td>
</tr>
<tr>
<td>color19</td>
<td>BLUEGREEN</td>
</tr>
<tr>
<td>color20</td>
<td>OLIVE</td>
</tr>
<tr>
<td>color21</td>
<td>BROWN</td>
</tr>
<tr>
<td>color22</td>
<td>DKBLUE</td>
</tr>
<tr>
<td>color23</td>
<td>BLUE</td>
</tr>
</tbody>
</table>
Examples:
change color1 with

**ActiveX / VCL**

```plaintext
Doc.SetRTFColor(1, RGB(10, 10, 10))
```

**.NET**

```plaintext
Doc.SetRTFColor(1, Color.FromArgb(10, 10, 10))
```

Or you can add as many colors to the table as you like, for example:

**ActiveX / VCL**

```plaintext
Doc.SetRTFColor(31, COLOR_GREEN)
```

**.NET**

```plaintext
Doc.SetRTFColor(31, Doc.COLOR_GREEN)
or
Doc.SetRTFColor(31, Color.Green)
```

**See also:**

18.10 Build-In Paragraph Settings

VPE RTF has build-in a predefined paragraph setting. Note, whilst in RTF text the values for paragraph settings are in twips (= 1 / 1440 inch), all values for the properties and methods are specified in metric or inch units (depending on the setting for the Unit Transformation).

The following sections are a list of properties and methods to access those settings and their default values:

See also:

18.11 FirstIndent

RTF: First-Line indent. Specifies the left indent of the first line of a new paragraph.

**property VpeCoord VPE.FirstIndent**

- write; runtime only

**Possible Values:**
- the left indent of the first line

**Default:**
- 0

**Remarks:**
- The left indent is relative to the left border of the object's rectangle. For example:
  Doc.FirstIndent = 2 means, that the left margin is set to:
  \[ \text{left border of object rectangle} + 2 \text{ cm} \]

- The first-line indent and the left indent are used together (they are added) for the first line. To gain a hanging indent, specify for the first-line indent the negative value of the left indent, which accumulates to zero then.

**Example:**

```
Doc.FirstIndent = -1
Doc.LeftIndent = 1
```

Will result in a paragraph formatting like this:

This is the first line and
  this is the second line and
  this is the third line

**See also:**

18.12 LeftIndent

RTF: Left indent. Specifies the left indent of all lines of a new paragraph.

property VpeCoord VPE.LeftIndent

   write; runtime only

Possible Values:
   the left indent

Default:
   0

Remarks:
   The left indent is relative to the left border of the object's rectangle. For example:
   Doc.LeftIndent = 2 means, that the left margin is set to:
   [left border of object rectangle] + 2 cm

   The first-line indent and the left indent are used together (they are added) for the first line. To
   gain a hanging indent, specify for the first-line indent the negative value of the left indent,
   which accumulates to zero then.

Example:

   Doc.FirstIndent = -1
   Doc.LeftIndent = 1

   Will result in a paragraph formatting like this:

   This is the first line and
   this is the second line and
   this is the third line

See also:
18.13 RightIndent

RTF: Right indent. Specifies the right indent of all lines of a new paragraph.

property VpeCoord VPE.RightIndent

write; runtime only

Possible Values: the right indent

Default: 0

Remarks: The right indent is relative to the right border of the object's rectangle. For example:
Doc.RightIndent = 2 means, that the right margin is set to:
[right border of object rectangle] - 2 cm

Example:

```
Doc.RightIndent = 2
```

See also: "RTF - Rich Text Format" in the Programmer's Manual
18.14 SpaceBefore

The space before a new RTF paragraph. Note that VPE does intentionally NOT use this value for the first paragraph of a new object and additionally after a \page, \pagebb or \sect keyword - this means: it is also not used on a new page after an Auto Break has occurred (as normal RTF editors do).

**property VpeCoord VPE.SpaceBefore**

write; runtime only

**Possible Values:**
the space before a new paragraph

**Default:**
0

**See also:**
18.15 SpaceAfter

The space after a RTF paragraph. Note that VPE does intentionally NOT use this value for the last paragraph of an object and if a \page, \pagebb or \sect keyword follows - this means: before an Auto Break occurs. Also note that most RTF editors emit a \par command after the very last line of a document / section. This is not ignored by VPE. You will recognize it as a gap behind the last line, especially if you draw a surrounding box.

**property VpeCoord VPE.SpaceAfter**

- write; runtime only

**Possible Values:**
- the space after a paragraph

**Default:**
- 0

**See also:**
18.16 SpaceBetween

RTF: Space between lines; if 0 is specified, the line spacing is automatically determined by the
tallest character in the line; else this size is used only if it is taller than the tallest character.

property VpeCoord VPE.SpaceBetween

    write; runtime only

Possible Values:
the space between paragraphs

Default:
0

See also:
18.17 DefaultTabSize

This is the RTF default advance width, if a \tab is processed. The default tab size is only used in case that no individual tab is defined for the next tab position.

**property VpeCoord VPE.DefaultTabSize**

- write; runtime only

**Possible Values:**
- the default TAB advance width

**Default:**
- 1.25 (= 1.25 cm)

**See also:**
18.18 SetTab

Sets an individual RTF tab position. You may specify as many tab positions as you like.

**method void VPE.SetTab(**
  VpeCoord TabPosition,
  long Reserved
**)

*VpeCoord TabPosition*
  tab position

*long Reserved*
  for future use and should be set to zero

**Example:**

```
Doc.SetTab(1)
Doc.SetTab(2)
Doc.SetTab(5)
```

Sets three individual tab positions at 1cm, 2cm and 5cm

```
Doc.ClearTab(2)
```

Removes the tab-stop at position 2cm (which was previously set by SetTab()).

**See also:**

18.19 ClearTab

Removes an individual RTF tab position.

**method void VPE.ClearTab**

```
    VpeCoord TabPosition

)
```

**VpeCoord TabPosition**

tab position

**Example:**

```
Doc.SetTab(1)
Doc.SetTab(2)
Doc.SetTab(5)
```

Sets three individual tab positions at 1cm, 2cm and 5cm

```
Doc.ClearTab(2)
```

Removes the tab-stop at position 2cm (which was previously set by `SetTab()`).

**See also:**

18.20 ClearAllTabs

Removes all RTF tab settings, which had been set previously by SetTab().

method void VPE.ClearAllTabs()

See also:
18.21 ResetParagraph
Resets all RTF paragraph settings to their default values.

method void VPE.ResetParagraph()

See also:
18.22 Build-In Paragraph Settings: RTF Auto Page Break

VPE processes the three most important RTF paragraph styles (described in the next sections) to have the best control over text formatting when automatic page breaks occur.

Auto Page Breaks only occur if the property AutoBreakMode is set to AUTO_BREAK_ON or AUTO_BREAK_FULL.

The RTF properties KeepLines, KeepNextParagraph and ParagraphControl do not work between different RTF objects. They are only in effect within one and the same RTF object.

Example: You insert an RTF object and below a second one which is too long for the current page, so an AutoBreak is fired. In such a case the second RTF object is treated independently of the first one and paragraph control between the two objects is not in effect.

See also:

18.23 KeepLines

Keep RTF paragraph intact. A page break may not occur between the lines of a paragraph. Instead the whole paragraph is moved to the next page.

**property boolean VPE.KeepLines**

write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes</td>
</tr>
<tr>
<td>False</td>
<td>no</td>
</tr>
</tbody>
</table>

**Default:**

False

**Remarks:**

Page breaks may occur between the lines of paragraphs, e.g. at their exact borders (see [KeepNextParagraph](#)).

KeepLines, KeepNextParagraph and [ParagraphControl](#) may be used together.

**See also:**

18.24 KeepNextParagraph

Keep RTF paragraph with the next paragraph. A page break may not occur between the following paragraphs. This means, one paragraph may not stand alone at the top of a new page. Instead the paragraphs are moved to the next page.

property boolean VPE.KeepNextParagraph

write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes</td>
</tr>
<tr>
<td>False</td>
<td>no</td>
</tr>
</tbody>
</table>

Default:
False

Remarks:
Note: Page breaks may occur between the lines of paragraphs (see KeepLines).

KeepLines, KeepNextParagraph and ParagraphControl may be used together.

See also:
18.25 ParagraphControl

Keep RTF paragraph intact, a single line may not remain on the current page or on the next page. A page break may not occur for a following paragraph, if only the first line of the paragraph would remain on the current page, or if only the last line of the paragraph would be placed on the next page. The whole paragraph is moved to the next page instead. KeepLines, KeepNextPar and ParControl may be used together.

property boolean VPE.ParagraphControl

write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes</td>
</tr>
<tr>
<td>False</td>
<td>no</td>
</tr>
</tbody>
</table>

Default:
False

Remarks:
KeepLines, KeepNextParagraph and ParagraphControl may be used together.

See also:
19 Clickable Objects

19.1 Clickable Object

[GUI Control Only, Professional Edition and above]

Objects can be made clickable by assigning them a unique Object ID. Moving with the mouse over such an object in the preview changes the cursor to a pointing hand. If the user clicks onto such an object, VPE fires the event `DoObjectClicked()` (VCL: `OnObjectClicked()`, .NET: `ObjectClicked Event`) to your application which includes the assigned Object ID.

In reaction to the event you can for example open a separate dialog, showing more detailed information about the clicked text or image.

See also the source code of the "Clickable Objects" demo.
19.2 EnableClickEvents

[GUI Control Only, Professional Edition and above]

Specifies, whether clickable objects shall be activated (= be clickable) or not. This property is valid for the whole document. If you set it to False, no object in the preview will be clickable, regardless whether you assigned an Object ID to an object or not.

**property boolean VPE.EnableClickEvents**

write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>enable</td>
</tr>
<tr>
<td>False</td>
<td>disable</td>
</tr>
</tbody>
</table>

**Default:**

True
19.3 ObjectID

[GUI Control Only, Professional Edition and above]

Defines an Object ID that will be assigned to the next object(s), that are inserted into the
document. With the Object ID you are able to identify a clicked object later when showing the
preview. When the user clicks onto such an object with an assigned ID, the DoObjectClicked
(VCL: OnObjectClicked, .NET: ObjectClicked) event is sent together with the Object ID to your
application.

The Object ID may have any value that can be represented with a long integer. The Object ID has
to be different from 0 (zero). Setting the Object ID to zero means, that the object is not clickable.

property long VPE.ObjectID

read / write; runtime only

Possible Values:
an Object ID to identify the clicked object when processing the DoObjectClicked() event (VCL:
OnObjectClicked(), .NET: ObjectClicked())

Default:
0

Remarks:
You may assign one and the same Object ID to any number of objects. It is the whole
responsibility of your application, how it will work with Object ID's.

The following objects can be assigned an Object ID (and therefore can be made clickable):

- Box
- Ellipse
- Pie
- Print(-Box)
- Write(-Box)
- Picture
- Barcodes (1D and 2D)
- RTF
- Charts
- UDO
- FormField

Remarks:
You can read the value of this property. It is only accessible for reading while your application
is processing the event DoObjectClicked() (VCL: OnObjectClicked(), .NET: ObjectClicked()).
Reading this property while not processing the event will return invalid - and therefore useless
- data.

Example:

```
Doc.ObjectID = 1
Doc.Print(1, 1, "Monthly Report")
```

Assigns the ObjectID 1 to the text object "Monthly Report".
Assigns the ObjectID 2 to the image "data.bmp".

Assigns NO ObjectID to the text object "This image shows data.". Therefore this object is not clickable.

Processing of the DoObjectClicked() event in Visual Basic:

```vbnet
Private Sub Doc_DoObjectClicked(ByVal ObjectID As Long)
    MsgBox "Object #" + Str(ObjectID) + " was clicked"
End Sub
```

The event handler in the example above will show a message box with the Object ID of the clicked object.
19.4 ClickedObject

[GUI Control Only, Enterprise Edition and above]

Returns the VPE Object - instead of the ObjectID, which is only an integer - of the object that fired the DoObjectClicked (VCL: OnObjectClicked, .NET: ObjectClicked) event.

property TVPEObject VPE.ClickedObject

   read; runtime only

Remarks:
This property is only accessible while your application is processing the DoObjectClicked() (VCL: OnObjectClicked(), .NET: ObjectClicked()) event.

   Reading this property while not processing the event, will throw an exception.

See also:
"Important Note for VPE-VCL Users" in the Programmer's Manual
20 UDO - User Defined Objects

20.1 UDO - User Defined Objects

[Professional Edition and above, not available for Java, PHP, Python, Ruby]

With User Defined Objects it is possible to draw any kind of object within VPE - and therefore to preview and print it!

A User Defined Object is first of all the same like a Box-Object (see "The Object-Oriented Style" in the Programmer's Manual). It inherits all properties of the Box object, including BkgMode, PenSize, PenStyle, etc.

UDO's are event driven, that means: VPE sends an event to your application when the object needs to be painted on the output device (i.e. the preview, the printer, the fax or whatsoever).

Therefore you specify additionally a long integer value (named lParam) when creating a UDO, which will help you identifying the object later during the paint event.

At the moment VPE paints the UDO, VPE first of all draws the Box object. Afterwards VPE fires the event DoUDOPaint() (VCL: OnUDOPaint(), .NET: UDOPaint Event). In that moment, your application can access the device context of VPE to draw everything you want into the rectangle of the UDO!

Moreover, with the ActiveX you have the powerful property UDOPicture, which can be assigned an OLE / COM object so it is painted automatically into the rectangle of the UDO!
20.2 CreateUDO

[Professional Edition and above, not available for Java, PHP, Python, Ruby]

Creates a User Defined Object (UDO). lParam can be of any value and is for your private use to identify the object later during processing the DoUDOPaint() (VCL: OnUDOPaint(), .NET: UDOPaint()).

method void VPE.CreateUDO(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    long lParam
)  

VpeCoord Left, Top, Right, Bottom
    position and dimensions of the UDO

long lParam
    numeric ID to identify the UDO during the DoUDOPaint() event (VCL: OnUDOPaint(), .NET: UDOPaint())

Example:
    see UDODC

Remarks:
    Special care must be taken, when creating VPE Document files that contain pictures or UDO's. For details please see WriteDoc.

    VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.
20.3 UDOPicture

[Professional Edition and above, ActiveX only]

This property is only accessible while you are processing the event DoUDOPaint() (VCL: OnUDOPaint(), .NET: UDOPaint()). Accessing this property while not processing the event will return invalid - and therefore useless - data.

property IPictureDisp VPE.UDOPicture

write; runtime only

Possible Values:
Picture datatype offered by OLE / COM objects (of type IPictureDisp)

Example:
ActiveX - Painting an OLE Object with UDO (example in Visual Basic):

First, insert an OLE Control named OLE1 into a form. For example a Microsoft Excel Spreadsheet or a Corel Draw object.

Then enter the following code:

```vbnet
Private Sub Form_Load()
    rem Make sure, the server application is running by executing the
    rem following code:
    OLE1.DoVerb (vbOLEShow)
    OLE1.Visible = False
    Doc.OpenDoc
    Call Doc.CreateUDO(1, 1, -18, -18, 1)
    Call Doc.Preview
End Sub

Private Sub Doc_DoUDOPaint ()
    If Doc.UDO1Param = 1 Then
        Doc.UDOPicture = OLE1.Picture
    End If
End Sub
```
20.4 UDOIParam

[Professional Edition and above, not available for Java, PHP, Python, Ruby]

Sets / returns the long integer value lParam from a UDO.

property long VPE.UDOIParam

read; runtime only
also supported by TVPEObject: read / write; runtime only

Returns:
the id of the UDO that needs to be drawn

Remarks:
Only in case you are using the property of the VPE Document Object (ActiveX: VpeControl and VCL: TVPEngine): This property is only accessible while you are processing the event DoUDOPaint() (VCL: OnUDOPaint(), .NET: UDOPaint()). Reading this property while not processing the event will return invalid - and therefore useless - data. It returns the long integer value lParam you had specified when creating the UDO (by calling CreateUDO()). The value returned should be used to identify the object which needs to be drawn.

Example:
see UDODC
20.5 UDOUC

[Professional Edition and above, not available for Java, PHP, Python, Ruby. Not available for .NET, see UDOGraphics]

This property is only accessible while you are processing the event DoUDOPaint() (VCL: OnUDOPaint(), .NET: UDOPaint()). Retrieving this value while not processing the event will return invalid - and therefore useless - data.

It returns the current Device Context VPE is painting on (this maybe the screen or a printer Device Context). Using this Device Context, you can call any Windows GDI function and paint yourself into Device Context. Be careful, do not destroy the Device Context and make sure to free all used system resources. In case of misuse the system may hang or fail to work correctly.

**property long VPE.UDODC**

- **read; runtime only**

**Returns:**
the HDC of the UDO that has to be painted

**Remarks:**
Before VPE fires the UDOPaint - event, it draws the Box object the UDO object is inherited from. Afterwards VPE saves the Device Context with the Windows GDI function SaveDC() and creates a clipping rectangle around the object.

After you have finished painting and return control to VPE by returning from your event-handler, VPE will delete the clipping rectangle and restore the Device Context with the Windows GDI function RestoreDC().

**Example:**

```vb
Doc.CreateUDO(1, 1, -18, -18, 1);
```

Creates a UDO with the current properties of the Box-Object and the ID (=IParam) "1"

Processing of the DoUDOPaint() event in Visual Basic:

```vb
Doc.CreateUDO(1, 1, -18, -18, 1);
```

Creating a UDO with:

```vb
Private Sub Form_Load()
    Doc.OpenDoc
    Call Doc.CreateUDO(1, 1, -18, -18, 1)
    Call Doc.Preview
End Sub
```
The event handler itself:

```vba
Private Sub Doc_DoUDOPaint()
    rem Windows GDI LineTo() draws a line from the current position up to,
    rem but not including, the specified point.
    rem Therefore we add 1 to each coordinate.

    If Doc.UDO1Param = 1 Then
        Call MoveToEx(Doc.UDODC, Doc.nUDOLeft, Doc.nUDOTop, 0)
        Call LineTo(Doc.UDODC, Doc.nUDORight + 1, Doc.nUDOBottom + 1)
        Call MoveToEx(Doc.UDODC, Doc.nUDORight, Doc.nUDOTop, 0)
        Call LineTo(Doc.UDODC, Doc.nUDOLeft - 1, Doc.nUDOBottom + 1)
    End If
End Sub
```

The example draws two crossing lines within the UDO.
20.6 UDOGraphics

[.NET only]

This property is only accessible while you are processing the event DoUDOPaint() (VCL: OnUDOPaint(), .NET: UDOPaint()). Reading this property while not processing the event will return a null value.

It returns the current GDI+ drawing surface, which VPE is painting on (this maybe the screen or a printer). Using this GDI+ drawing surface, you can call any GDI+ method or property and paint yourself onto the drawing surface.

property Graphics VPE.UDOGraphics

   read; runtime only

Returns:
   the Graphics object of the UDO that has to be painted

Remarks:
   Before VPE fires the UDOPaint() event, it draws the Box object the UDO object is inherited from.

Example:

```
Doc.CreateUDO(1, 1, -18, -18, 1);

Creates a UDO with the current properties of the Box-Object and the ID (=lParam) "1"
```

Processing of the UDOPaint() event in C#: creating UDO's

```
private void Form1_Load(object sender, System.EventArgs e)
{
    Doc.OpenDoc();
    Doc.PenSize = 0;       // no frame
    Doc.CreateUDO(1, 1, -11.5, -1, 1);
    Doc.CreateUDO(1, 3, -11.5, -1, 2);
    Doc.Preview();
}
```
The event handler for painting the content of the UDO:

```csharp
private void Doc_UDOPaint(object sender, System.EventArgs e)
{
    int left = Doc.nUDOLeft;
    int top = Doc.nUDOTop;
    int width = Doc.nUDORight - vpe.nUDOLeft;
    int height = Doc.nUDOBOTTOM - vpe.nUDOTop;

    Doc.UDOGraphics.DrawRectangle(
        new Pen(Color.Green, 3), left, top, width, height);

    if (Doc.UDOLParam == 1)
    {
        Doc.UDOGraphics.DrawString(
            "Hello, I am a UDO (User Defined Object).",
            new Font("Arial", 10, FontStyle.Bold),
            new SolidBrush(Color.Black),
            left, top);
    }
    else if (Doc.UDOLParam == 2)
    {
        Doc.UDOGraphics.DrawString(
            "Hello, I am UDO #2."
            new Font("Arial", 12, FontStyle.Italic),
            new SolidBrush(Color.Green),
            left, top);
    }
}
```

The example draws the contents of two different UDO's. You will notice that the same
UDOPaint() event handler is called for each UDO that appears in the document. The UDO that
needs to be drawn is identified by the property UDOlParam. When we had called CreateUDO()
we had specified for the first UDO for UDOlParam the value 1, and for the second the value 2.
20.7 UDOIsPrinting

[Professional Edition and above, not available for Java, PHP, Python, Ruby]

If you want to be able, to scale the visual content of a UDO accordingly to the scale of the Preview, the scale of a printing device or the scale of a virtual surface (i.e. when exporting to PDF or an image) you need to be able to determine onto what surface the UDO has to be drawn. Use this property as well as the UDOIsExporting property for such a task. Do not use the property IsPrinting because while the print job is running and IsPrinting = true, it can still happen that the UDO needs to be painted onto the screen surface, for example because you moved a window of another application over the preview of VPE.

property boolean VPE.UDOIsPrinting

read; runtime only

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, the UDO has to be drawn onto a printer’s surface</td>
</tr>
<tr>
<td>False</td>
<td>no, the UDO has not to be drawn onto a printer’s surface</td>
</tr>
</tbody>
</table>

Remarks:
This property is useful for the ActiveX and VCL. The .NET control sets the UDOGraphics.ScaleTransform() already accordingly to the output surface and any applied scaling factor.
20.8 UDOIsExporting

[Professional Edition and above, not available for Java, PHP, Python, Ruby]

If you want to be able, to scale the visual content of a UDO accordingly to the scale of the Preview, the scale of a printing device or the scale of a virtual surface (i.e. when exporting to PDF or an image) you need to be able to determine onto what surface the UDO has to be drawn. Use this property as well as the UDOIsPrinting property for such a task.

**property boolean VPE.UDOIsExporting**

read; runtime only

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, the UDO has to be drawn onto a virtual surface</td>
</tr>
<tr>
<td>False</td>
<td>no, the UDO has not to be drawn onto a virtual surface</td>
</tr>
</tbody>
</table>

**Remarks:**
This property is useful for the ActiveX and VCL. The .NET control sets the UDOGraphics.ScaleTransform() already accordingly to the output surface and any applied scaling factor.
20.9 UDODpiX

[Professional Edition and above, not available for Java, PHP, Python, Ruby]

Returns the X-resolution of the UDO’s current output surface in DPI (Dots Per Inch).

property boolean VPE.UDODpiX

read; runtime only

Returns:
The X-resolution of the UDO’s current output surface in DPI (Dots Per Inch).

Remarks:
This property is useful for the ActiveX and VCL. The .NET control sets the
UDOGraphics.ScaleTransform() already accordingly to the output resolution.
20.10 UDODpiY

[Professional Edition and above, not available for Java, PHP, Python, Ruby]
Returns the Y-resolution of the UDO’s current output surface in DPI (Dots Per Inch).

property boolean VPE.UDODpiY

read; runtime only

Returns:
The Y-resolution of the UDO’s current output surface in DPI (Dots Per Inch).

Remarks:
This property is useful for the ActiveX and VCL. The .NET control sets the
UDOGraphics.ScaleTransform() already accordingly to the output resolution.
20.11 nUDOLeft, nUDOTop, nUDORight, nUDOBottom

[Professional Edition and above, not available for Java, PHP, Python, Ruby]

Return each coordinate of the UDO rectangle (this is exactly the area you may paint in).

The properties are only accessible while you are processing the event `DoUDOPaint()` (VCL: `OnUDOPaint()`, .NET: `UDOPaint()`). Retrieving the values while not processing the event will return invalid - and therefore useless - data.

```plaintext
property long VPE.nUDOLeft
property long VPE.nUDOTop
property long VPE.nUDORight
property long VPE.nUDOBottom
```

read; runtime only

**Returns:**
the coordinates of the UDO which fired the currently processed UDO-Paint-Event

**Remarks:**
In contrast to the other n-Properties the nUDO-Properties return the bounding rectangle of the UDO in **DEVICE COORDINATES** (!) (pixels, not metric units). Therefore, independently from the resolution of the output device or the scaling factor of the preview, the coordinates are computed by VPE accordingly. Be it the 96 DPI screen, a 200 DPI fax, a 600 DPI printer or a 300 DPI export image (yes, UDO works with PictureExport, too)!
21 Picture Export Functions

21.1 Picture Export Functions

[Windows Platform Only, Professional Edition and above]

VPE is able to export pages or rectangular parts of pages from the document to image files.

The supported formats for Picture Export are:

- BMP
- WMF (Windows only)
- EMF (Windows only)
- JPEG (compression ratio can be set freely)
- PNG
- TIFF 6.0 (Fax G3, Fax G4, LZW, Packbits, Deflate, JPEG, Multipage)
- GIF (Multipage)

For all bitmap formats you can specify the color depth and the resolution (in DPI). Additionally dithering is possible.

Exported Metafiles:

Text justification and included bitmaps may not be correctly displayed, if you re-import the generated metafiles into the preview, because exported metafiles are always rendered to a virtual 600 DPI resolution. So displaying them on the 96 DPI screen brings tolerances into the object placement due to coordinate rounding problems. Nevertheless if you print them on 300 DPI printers, or a multiple of 300 DPI printers (600 DPI, 1200 DPI, etc.), they are printed perfectly.

The following restrictions apply only to exported WMF files:

- **Barcodes** are not exported
- Metafiles inside the document are not exported
- Some software does not rely onto the information provided in the header of the WMF about its dimensions, instead it scans the WMF file to compute its dimensions. This can cause problems, if a part of a page is only exported, when objects which intersect the part are also outside this area. The computing software will include the whole object and distort the WMF. This is a problem of the software, which tries to be too clever.
21.2 **JpegExportOptions**

[Windows Platform Only, Professional Edition and above]

Specifies options for exported JPEG images.

**property JpegExportOptions [long] VPE.JpegExportOptions**

read, write; runtime only

**Possible Values:**
several options

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICEXP_JPEG_DEFAULT</td>
<td>0</td>
<td>Default</td>
<td>Default, saves with good quality (75:1)</td>
</tr>
<tr>
<td>PICEXP_JPEG_HI_QUALITY</td>
<td>128</td>
<td>HiQuality</td>
<td>Saves with superb quality (100:1)</td>
</tr>
<tr>
<td>PICEXP_JPEG_GOOD_QUALITY</td>
<td>256</td>
<td>GoodQuality</td>
<td>Saves with good quality (75:1)</td>
</tr>
<tr>
<td>PICEXP_JPEG_MID_QUALITY</td>
<td>512</td>
<td>MidQuality</td>
<td>Saves with normal quality (50:1)</td>
</tr>
<tr>
<td>PICEXP_JPEG_LO_QUALITY</td>
<td>1024</td>
<td>LoQuality</td>
<td>Saves with average quality (25:1)</td>
</tr>
<tr>
<td>PICEXP_JPEG_BAD_QUALITY</td>
<td>2048</td>
<td>BadQuality</td>
<td>Saves with bad quality (10:1)</td>
</tr>
<tr>
<td>Integer X in [0..100]</td>
<td>0 – 100</td>
<td>n/a</td>
<td>Save with quality X:1</td>
</tr>
<tr>
<td>PICEXP_JPEG_PROGRESSIVE</td>
<td>8192</td>
<td>Progressive</td>
<td>If set, JPG files are written progressive (interlaced)</td>
</tr>
</tbody>
</table>

**Default:**
PICEXP_JPEG_DEFAULT

**Remarks:**
The flag PICEXP_JPEG_PROGRESSIVE can be combined with any of the other flags.

**Example:**

```
Doc.JpegExportOptions = PICEXP_JPEG_DEFAULT + PICEXP_JPEG_PROGRESSIVE
```
21.3 TiffExportOptions

[Windows Platform Only, Professional Edition and above]

Specifies options for exported TIFF images.

**property TiffExportOptions [long] VPE.TiffExportOptions**

read, write; runtime only

**Possible Values:**
several options

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICEXP_TIFF_DEFAULT</td>
<td>0</td>
<td>Default</td>
<td>Default, save using CCITTFAX4 compression for 1-bit bitmaps and LZW compression for any other bitmaps</td>
</tr>
<tr>
<td>PICEXP_TIFF_PACKBITS</td>
<td>256</td>
<td>Packbits</td>
<td>Save using PACKBITS compression</td>
</tr>
<tr>
<td>PICEXP_TIFF_DEFLATE</td>
<td>512</td>
<td>Deflate</td>
<td>Save using DEFLATE compression (a.k.a. ZLIB compression)</td>
</tr>
<tr>
<td>PICEXP_TIFF_ADOBE_DEFLATE</td>
<td>1024</td>
<td>AdobeDeflate</td>
<td>Save using ADOBE DEFLATE compression</td>
</tr>
<tr>
<td>PICEXP_TIFF_NONE</td>
<td>2048</td>
<td>None</td>
<td>Save without any compression</td>
</tr>
<tr>
<td>PICEXP_TIFF_CCITTFAX3</td>
<td>4096</td>
<td>Ccittfax3</td>
<td>Save using CCITT Group 3 fax encoding</td>
</tr>
<tr>
<td>PICEXP_TIFF_CCITTFAX4</td>
<td>8192</td>
<td>Ccittfax4</td>
<td>Save using CCITT Group 4 fax encoding</td>
</tr>
<tr>
<td>PICEXP_TIFF_LZW</td>
<td>16384</td>
<td>Lzw</td>
<td>Save using LZW compression</td>
</tr>
<tr>
<td>PICEXP_TIFF_JPEG</td>
<td>32768</td>
<td>Jpeg</td>
<td>Save using JPEG compression (8-bit greyscale and 24-bit only. Default to LZW for other bitdepths.)</td>
</tr>
<tr>
<td>PICEXP_TIFF_APPEND</td>
<td>1073741824</td>
<td>Append</td>
<td>Append file to multi-page TIFF</td>
</tr>
</tbody>
</table>

**Default:**
PICEXP_TIFF_DEFAULT

**Remarks:**
The flag PICEXP_TIFF_APPEND can be combined with other flags.

**Example:**

```
Doc.TiffExportOptions = PICEXP_TIFF_DEFAULT + PICEXP_TIFF_APPEND
```
21.4 BmpExportOptions

[Windows Platform Only, Professional Edition and above]

Specifies options for exported BMP images.

property BmpExportOptions [long] VPE.BmpExportOptions

   read, write; runtime only

Possible Values:
    several options

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICEXP_BMP_DEFAULT</td>
<td>0</td>
<td>Default</td>
<td>Default, save without any compression</td>
</tr>
<tr>
<td>PICEXP_BMP_RLE</td>
<td>1</td>
<td>Rle</td>
<td>Save RLE compressed</td>
</tr>
</tbody>
</table>

Default:
   PICEXP_BMP_DEFAULT

Example:

   Doc.BmpExportOptions = PICEXP_BMP_RLE
21.5 PnmExportOptions

[Windows Platform Only, Professional Edition and above]

Specifies options for exported PBM, PGM, PPM images.

property PnmExportOptions [long] VPE.PnmExportOptions

read, write; runtime only

Possible Values:
several options

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICEXP_PNM_DEFAULT</td>
<td>0</td>
<td>Default</td>
<td>PBM, PGM, PPM files are written RAW</td>
</tr>
<tr>
<td>PICEXP_PNM_ASCII</td>
<td>1</td>
<td>Ascii</td>
<td>PBM, PGM, PPM files are written ASCII</td>
</tr>
</tbody>
</table>

Default:
PICEXP_PNM_DEFAULT

Example:

```c
Doc.PnmExportOptions = PICEXP_PNM_ASCII
```
21.6 GifExportOptions

[Windows Platform Only, Professional Edition and above]

Specifies options for exported GIF images.

**property GifExportOptions [long] VPE.GifExportOptions**

  read, write; runtime only

**Possible Values:**
  several options

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICEXP_GIF_DEFAULT</td>
<td>0</td>
<td>Default</td>
<td>nothing special</td>
</tr>
<tr>
<td>PICEXP_GIF_APPEND</td>
<td>1073741824</td>
<td>Append</td>
<td>append file to multi-page GIF</td>
</tr>
</tbody>
</table>

**Default:**

PICEXP_GIF_DEFAULT

**Remarks:**

The flag PICEXP_GIF_APPEND can be combined with any of the other flags.

**Example:**

```plaintext
Doc.GifExportOptions = PICEXP_GIF_DEFAULT + PICEXP_GIF_APPEND
```
21.7 PictureExportColorDepth

[Windows Platform Only, Professional Edition and above]

Specifies the color depth of an exported image.

property long VPE.PictureExportColorDepth

read; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICEXP_COLOR_MONO</td>
<td>1</td>
<td>ColorMono</td>
<td></td>
</tr>
<tr>
<td>PICEXP_COLOR_16</td>
<td>4</td>
<td>Color16</td>
<td></td>
</tr>
<tr>
<td>PICEXP_COLOR_256</td>
<td>8</td>
<td>Color256</td>
<td></td>
</tr>
<tr>
<td>PICEXP_COLOR_HI</td>
<td>16</td>
<td>ColorHi</td>
<td></td>
</tr>
<tr>
<td>PICEXP_COLOR_TRUE</td>
<td>24</td>
<td>ColorTrue</td>
<td></td>
</tr>
</tbody>
</table>

Default:

PICEXP_COLOR_TRUE

Remarks:

The higher the color depth, the more memory is needed during export to create the image. The color depth can not be specified for metafiles (WMF / EMF), metafiles are always exported in true-color.

See also:

PictureExportDither
21.8 PictureExportDither

[Windows Platform Only, Professional Edition and above]

Specifies, if an exported image shall be dithered to a lower color resolution when written to file. This is very useful, if you want to export smaller images (regarding their size in bytes) - or if you want to fax (in b/w) a true-color image - and to keep the visual information at a high quality.

property long VPE.PictureExportDither

read; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICEXD_DITHER_NONE</td>
<td>0</td>
<td>DitherNone</td>
<td></td>
</tr>
<tr>
<td>PICEXD_DITHER_MONO</td>
<td>1</td>
<td>DitherMono</td>
<td></td>
</tr>
<tr>
<td>PICEXD_DITHER_256</td>
<td>3</td>
<td>Dither256</td>
<td></td>
</tr>
<tr>
<td>PICEXD_DITHER_256_WU</td>
<td>4</td>
<td>Dither256Wu</td>
<td></td>
</tr>
</tbody>
</table>

Default:

PICEXD_DITHER_NONE

Remarks:

For PICEXD_DITHER_MONO: images are dithered with the Floyd-Steinberg dithering algorithm. The source bitmap may have any of the following color depths: 1, 4, 8, 16, 24, 32 bits. If the source bitmap is a monochrome bitmap, VPE creates a copy without dithering.

For PICEXD_DITHER_256: this only works, if the source bitmap is 24-bit, i.e. `PictureExportColorDepth` is PICEXD_COLOR_TRUE. Images are not dithered, instead VPE uses color-reduction (NeuQuant neural-net quantization algorithm by Anthony Dekker). This algorithm creates very good results but is rather slow.

For PICEXD_DITHER_256_WU: this only works, if the source bitmap is 24-bit, i.e. `PictureExportColorDepth` is PICEXD_COLOR_TRUE. Images are not dithered, instead VPE uses color-reduction (Xiaolin Wu color quantization algorithm). This algorithm is much faster than the NeuQuant algorithm, but does not create as good results. In some cases it might lead to false colors.

Example:

**ActiveX / VCL:**

```vbnet
Doc.PictureExportColorDepth = PICEXD_COLOR_256
Doc.PictureExportDither = PICEXD_DITHER_MONO
Doc.PictureExportPage("test.bmp", 1)
```

**.NET:**

```csharp
Doc.PictureExportColorDepth = PictureExportColorDepth.Color256
Doc.PictureExportDither = PictureExportDither.DitherMono
Doc.PictureExportPage("test.bmp", 1)
```

Instructs VPE, to generate from page one of the document internally a 256 color image first, and to dither it down afterwards to b / w (monochrome) when writing it to file.
21.9 PictureExportPage

[Windows Platform Only, Professional Edition and above]

Exports the page specified in parameter "page_no" to file. The settings for the export options of the specific file type, PictureExportColorDepth and PictureExportDither are used.

The type of image is automatically determined by the suffix of FileName (e.g. ".JPG" = JPEG). If FileName contains no suffix or a suffix which does not specify an image type (e.g. ".001"), the image type can be specified with the property PictureType.

The resolution of the image is defined with DefaultPictureDPI().

method boolean VPE.PictureExportPage(
    string FileName,
    long PageNo
)

string FileName
   (path- and) filename of exported image

long PageNo
   page number of the page that shall be exported from the document

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success</td>
</tr>
<tr>
<td>False</td>
<td>failure</td>
</tr>
</tbody>
</table>

Remarks:

In case of an error, LastError is set.

During export, the BusyProgressBar is shown.

The higher the color depth and the higher the resolution, the more memory is needed during export to create the image. The color depth and resolution can not be specified for metafiles (WMF / EMF), metafiles are always exported in true-color with a resolution of 600 x 600 DPI.
Example:

**ActiveX / VCL:**

```
Doc.PictureExportColorDepth = PICEXP_COLOR_256
Doc.PictureExportDither = PICEXP_DITHER_MONO
Doc.SetDefaultPictureDPI(300, 300)
Doc.PictureExportPage("test.bmp", 1)
```

**.NET:**

```
Doc.PictureExportColorDepth = PictureExportColorDepth.Color256
Doc.PictureExportDither = PictureExportDither.DitherMono
Doc.SetDefaultPictureDPI(300, 300)
Doc.PictureExportPage("test.bmp", 1)
```

Instructs VPE, to generate from page one of the document internally a 256 color image with 300 x 300 DPI resolution first, and to dither it down afterwards to b/w (monochrome) in the same 300 x 300 DPI resolution when writing it to a BMP file named "test.bmp".

Example 2:

**ActiveX / VCL:**

```
// Export as True-Color image
Doc.PictureExportColorDepth = PICEXP_COLOR_TRUE

// VPE determines the file type from the filename suffix. Because we use
// here for the suffix ".001", ".002", etc., we tell VPE the type of file:
Doc.PictureType = PIC_TYPE_PNG

// PNG is always written with Deflate (ZLib) compression

// Set 96 DPI resolution:
Doc.SetDefaultPictureDPI(96, 96)
Doc.PictureExportPage("image.001", 2);
```

**.NET:**

```
// Export as True-Color image
Doc.PictureExportColorDepth = PictureExportColorDepth.ColorTrue

// VPE determines the file type from the filename suffix. Because we use
// here for the suffix ".001", ".002", etc., we tell VPE the type of file:
Doc.PictureType = PictureType.PNG

// Select ZIP compression:
Doc.PictureExportOptions = PictureExportOptions.WriteCompressed

// Set 96 DPI resolution:
Doc.SetDefaultPictureDPI(96, 96)
Doc.PictureExportPage("image.001", 2);
```

Instructs VPE, to export page 2 of the document as true-color Deflate-compressed PNG image with 96 x 96 DPI resolution named "image.001".
Example 3:

**ActiveX / VCL:**

```
// Export as dithered Black and White image:
Doc.PictureExportDither = PICEXP_DITHER_MONO

// Export as TIFF image:
Doc.PictureType = PIC_TYPE_TIFF

// Select FaxG4 as export format:
Doc.PictureExportOptions = PICEXP_TIFFFCITTFAX4

// Set 200 DPI resolution:
Doc.SetDefaultPictureDPI(200, 200)
Doc.PictureExportPage("image.002", 3)
```

**.NET:**

```
// Export as dithered Black and White image:
Doc.PictureExportDither = PictureExportDither.Mono

// Export as TIFF image:
Doc.PictureType = PictureType.TIFF

// Select FaxG4 as export format:
Doc.PictureExportOptions = PictureExportOptions.TIFF_CCITT_Fax4

// Set 200 DPI resolution:
Doc.SetDefaultPictureDPI(200, 200)
Doc.PictureExportPage("image.002", 3)
```

Instructs VPE, to export page 3 of the document as dithered b / w Fax G4 TIFF image with 200 x 200 DPI resolution named "image.002".
21.10 PictureExportPageStream

[Windows Platform Only, Professional Edition and above]
Identical to PictureExportPage(), but exports the picture to a stream.

```java
method boolean VPE.PictureExportPageStream(
    TVPEStream stream,
    long PageNo
)
```

`TVPEStream stream`
The stream-object where the picture is written to. The stream must have been created before by calling `CreateMemoryStream()`.

`long PageNo`
page number of the page that shall be exported from the document

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success</td>
</tr>
<tr>
<td>False</td>
<td>failure</td>
</tr>
</tbody>
</table>

Remarks:
WMF and EMF files are written to a temporary file internally first and afterwards to the stream. This is, because the Windows API does not allow to stream metafiles.
21.11 PictureExport

[Windows Platform Only, Professional Edition and above]

Exports a rectangular part specified by x, y, x2, y2 of the page specified in parameter "page_no" to file. The settings for the export options of the specific file type, PictureExportColorDepth and PictureExportDither are used.

The type of image is automatically determined by the suffix of FileName (e.g. ".JPG" = JPEG). If FileName contains no suffix or a suffix which does not specify an image type (e.g. ".001"), the image type can be specified with the property PictureType.

The resolution of the image is defined with DefaultPictureDPI().

```csharp
method boolean VPE.PictureExport(
    string FileName,
    long PageNo,
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom
)
```

- **string FileName**
  (path- and) filename of exported image

- **long PageNo**
  page number of the page that shall be exported from the document

- **VpeCoord Left, Top, Right, Bottom**
  rectangle of the page in metric or inch units that shall be exported as image

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success</td>
</tr>
<tr>
<td>False</td>
<td>failure</td>
</tr>
</tbody>
</table>

**Remarks:**

In case of an error, LastError is set.

During export, the BusyProgressBar is shown.

For Left, Top, Right, Bottom you can also specify the V-Flags and for Right, Bottom negative values to specify a delta value instead of an absolute coordinate.

The higher the color depth and the higher the resolution, the more memory is needed during export to create the image.

Metafiles are only supported on the Windows platform. The color depth and resolution can not be specified for metafiles (WMF / EMF), metafiles are always exported in true-color with a resolution of 600 x 600 DPI. The maximum dimensions for exported WMF is limited to 60 x 60 cm, this does not apply to EMF.
Example:

**ActiveX / VCL:**

```plaintext
Doc.PictureExportColorDepth = PICEXP_COLOR_256
Doc.PictureExportDither = PICEXP_DITHER_MONO
Doc.SetDefaultPictureDPI(300, 300)
Doc.PictureExport("test.bmp", 1, 1, -5, -5)
```

**.NET:**

```plaintext
Doc.PictureExportColorDepth = PictureExportColorDepth.Color256
Doc.PictureExportDither = PictureExportDither.DitherMono
Doc.SetDefaultPictureDPI(300, 300)
Doc.PictureExport("test.bmp", 1, 1, -5, -5)
```

Instructs VPE to generate from the rectangle (1, 1, 6, 6) from page one of the document internally a 256 color image with 300 x 300 DPI resolution first, and to dither it down afterwards to b/w (monochrome) in the same 300 x 300 DPI resolution when writing it to the BMP file named "test.bmp".

**ActiveX / VCL:**

```plaintext
Doc.PictureExport("test.bmp", 1, VLEFT, VTOP, VRIGHT, VBOTTOM)
```

**.NET:**

```plaintext
Doc.PictureExport("test.bmp", 1, Doc.nLeft, Doc.nTop, Doc.nRight, Doc.nBottom)
```

Instructs VPE to export the last inserted object to an image file.
21.12 PictureExportStream

[Windows Platform Only, Professional Edition and above]

Identical to `PictureExport()`, but exports the picture to a stream.

```c
method int VPE.PictureExportPageStream(
    TVPEStream stream,
    long PageNo,
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom
)
```

**TVPEStream stream**
- The stream-object where the picture is written to. The stream must have been created before by calling `CreateMemoryStream()`.

**long PageNo**
- page number of the page that shall be exported from the document

**VpeCoord Left, Top, Right, Bottom**
- rectangle of the page in metric or inch units that shall be exported as image

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success</td>
</tr>
<tr>
<td>False</td>
<td>failure</td>
</tr>
</tbody>
</table>

**Remarks:**
- WMF and EMF files are written to a temporary file internally first and afterwards to the stream. This is, because the Windows API does not allow to stream metafiles.
22 Memory Streams (ActiveX / VCL)

22.1 Memory Streams (ActiveX / VCL)

[Professional Edition and above]

This chapter explains special memory streams of VPE, which allow to read and write documents and images from/to memory. This is especially useful, if you want to store VPE documents or images in databases as BLOBs, or if you wish to create for example PDF documents on a web server in memory - without writing them to disk - in order to send them directly via HTTP to clients.

A memory stream is like a file, which is held in memory only.

This chapter describes the methods and functions for the ActiveX / VCL. For the other VPE controls (.NET, Java, PHP, etc.) a special TVPEStream class (see Memory Streams on page 600) has been implemented, which is derived from System.IO.Stream and is therefore compatible to the stream classes of .NET.

Background:

For export, the basic idea is to write a VPE (or PDF, HTML, etc.) document to a memory stream first, using WriteDocStream(stream). In order to be able to access the data of the memory stream (for sending it to a client's browser or to store it as BLOB in a database), you need to use TVPEStream.Read() afterwards. This will copy the data from the stream to a buffer of your application.

Note that you can also export pages as images like TIF, JPG, etc. to memory streams using equivalently the methods PictureExportStream() and PictureExportPageStream().

Vice versa for import, the basic idea is to write a VPE document as BLOB from a database (or from anywhere else, e.g. a network stream) to a memory stream first, using TVPEStream.Write(). This will copy the data from a buffer of your application to the stream. In order to import the document from the stream into VPE, you need to use ReadDocStream(stream) afterwards.

Note that you can also read (import) images from memory streams using equivalently the methods PictureStream() or RenderPictureStream(). And you can also read (import) RTF (Rich Text) from memory streams using WriteRTFStream(), WriteBoxRTFStream(), RenderRTFStream() and RenderBoxRTFStream(). Please don’t be confused that the RTF-methods begin with the word "write", this will not write to the stream, but create the RTF objects from the given stream. The names have been chosen accordingly to the RTF-methods which create RTF objects from strings.
Important:

After any write or read operation to a memory stream, the position of the internal memory stream pointer is, where the last read or write operation did stop. E.g. after calling WriteDocStream(), a ReadDocStream() on the written stream will fail. You must call Seek(0) first, to position the internal memory stream pointer at the beginning of the memory stream, before executing ReadDocStream().

The same is true, if you wish to read data from a memory stream using TVPEStream.Read() – where the memory stream has been filled before by WriteDocStream(). Call Seek(0) first!

This applies to all read and write operations, i.e. PictureStream(), etc.

Example:

```pascal
Vpe.OpenDoc()

// Write the current document to a memory stream
TVPEStream Stream = Vpe.CreateMemoryStream(0)
Vpe.WriteDocStream(Stream)

// Create a second document
Vpe2.OpenDoc()

// Read the memory stream into the new document, seek to position 0 first!
Stream.Seek(0);
Vpe.ReadDocStream(Stream);

// Cleanup
Stream.Close();
```

See also:

- WriteDocStream
- WriteDocStreamPageRange
- ReadDocStream
- ReadDocStreamPageRange
- PictureStream
- RenderPictureStream
- PictureExportStream
- PictureExportPageStream
- WriteRTFStream
- WriteBoxRTFStream
- RenderRTFStream
- RenderBoxRTFStream
22.2 CreateMemoryStream (ActiveX / VCL)

[Professional Edition and above; ActiveX / VCL only]

Creates a new empty stream in memory. A memory stream can grow as large as you wish, the size is only limited by available memory. VPE divides a memory stream into chunks of equal size - by default 16 KB. This means that an initial stream only occupies 16 KB of memory. If you write more data to a memory stream, the memory stream is enlarged by additional chunks as required. Internally VPE stores each chunk separately and links each chunk with the next.

This is a method of the VPE document. All other methods and properties described throughout this chapter belong to the TVPEStream object, which is created by this method.

```csharp
method TVPEStream VPE.CreateMemoryStream(
    long chunk_size
)

long chunk_size
    The size of each chunk, if you set this parameter to zero, VPE uses a default chunk size of 16 KB, which is a reasonable value.

Returns:
    A TVPEStream object for the created memory stream.

In case of an error an exception is thrown and LastError is set.

You must ALWAYS close all streams you ever opened or created, when you have done using them. Not doing so will result in memory leaks (only at the moment your application is closed, all streams will be released from memory). Neither calling CloseDoc(), nor destroying a TVPEStream object will close a stream.

Returns:
    sets LastError
22.3 Close (ActiveX / VCL)
[Professional Edition and above; ActiveX / VCL only]
Closes the given stream.

method void TVPEStream.Close( )

Remarks:
When closing a stream, the internal stream handle is destroyed. You may NOT use this object in any further calls to a method, which takes a stream object as parameter.
22.4 Read (ActiveX / VCL)

[Professional Edition and above; ActiveX / VCL only]

Reads data from the supplied stream.

method int TVPEStream.Read(
    BYTE *buffer,
    int size
)

BYTE *buffer
    pointer to the buffer, to which the data will be copied from the stream

int size
    the number of bytes that shall be read

Returns:
    The number of bytes read from the stream.

Remarks:
    When reading from a stream, which was written by a VPE method, e.g. WriteDocStream() or PictureStream() etc., you need to call Seek(0) first before calling Read() in order to reset the internal stream pointer to the beginning of the stream.
22.5 Write (ActiveX / VCL)

[Professional Edition and above; ActiveX / VCL only]

Writes data to the supplied stream.

**method boolean TVPEStream.Write(***

BYTE *buffer,
int size
**)

* BYTE *buffer
  pointer to the buffer, from which the data will be copied to the stream

* int size
  the number of bytes that shall be written

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success</td>
</tr>
<tr>
<td>False</td>
<td>failure</td>
</tr>
</tbody>
</table>

**Remarks:**

If you want to call a VPE method (for example ReadDocStream() or PictureStream()), after you have written data to a stream, you need to call Seek(0) first before calling Write() in order to reset the internal stream pointer to the beginning of the stream.
22.6 Size (ActiveX / VCL)

[Professional Edition and above; ActiveX / VCL only]

Returns the size - in bytes - of the given stream.

property long TVPEStream.Size

    read; runtime only

Returns:
    The size of the stream in bytes.
22.7 IsEof (ActiveX / VCL)

[Professional Edition and above; ActiveX / VCL only]

Returns the End Of File status of the given stream.

This property returns true after the first read operation that attempts to read past the end of the file. IsEof continues to report true, until Seek() (see “Seek (ActiveX / VCL)” on page 597) is called.

property boolean TVPEStream.IsEof

    read; runtime only

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>EOF</td>
</tr>
<tr>
<td>False</td>
<td>not EOF</td>
</tr>
</tbody>
</table>
22.8 State (ActiveX / VCL)

[Professional Edition and above; ActiveX / VCL only]

Returns the status of the stream.

For memory streams the state can become false, if the system is out-of-memory, or if you try to read past the end of file, or if you seek outside the file.

**property boolean TVPEStream.State**

read; runtime only

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>status ok</td>
</tr>
<tr>
<td>False</td>
<td>failure</td>
</tr>
</tbody>
</table>
22.9 Position (ActiveX / VCL)

[Professional Edition and above; ActiveX / VCL only]

Returns the current position - in bytes - of the internal file pointer.

**property long TVPEStream.Position**

read; runtime only

**Returns:**

The current position within the stream.
22.10 Seek (ActiveX / VCL)

[Professional Edition and above; ActiveX / VCL only]

Moves the file pointer to the given position.

```c
method void TVPEStream.Seek(
    long pos
)
```

`long pos`
the new position
22.11 SeekEnd (ActiveX / VCL)

[Professional Edition and above; ActiveX / VCL only]

Moves the file pointer to the given position, which is relative to the end of the stream. Therefore the parameter `pos` must always be negative or null.

```c
method void TVPESStream.SeekEnd(
    long pos
)
```

`long pos`

the new position, must always be negative or null
22.12 SeekRel (ActiveX / VCL)

[Professional Edition and above; ActiveX / VCL only]

Moves the file pointer to the given position, which is relative to the current position.

**method void TVPEStream.SeekRel(  
   long offset
)

long pos  
the new position, relative to the current position**
23 Memory Streams

23.1 Memory Streams

[Professional Edition and above; .NET, Java, PHP, etc.]

This chapter explains special memory streams of VPE, which allow to read and write documents and images from / to memory. This is especially useful, if you want to store VPE documents or images in databases as BLOBs, or if you wish to create for example PDF documents on a web server in memory - without writing them to disk - in order to send them directly via HTTP to clients.

A memory stream is like a file, which is held in memory only.

This chapter describes the methods and functions for the VPE .NET component, where the stream class has been derived from System.IO.Stream and is therefore compatible to the stream classes of .NET.

Important:

After any write or read operation to a memory stream, the position of the internal memory stream pointer is, where the last read or write operation did stop. E.g. after calling WriteDocStream(), a ReadDocStream() on the written stream will fail. You must call Seek(0) first, to position the internal memory stream pointer at the beginning of the memory stream, before executing ReadDocStream().

The same is true, if you wish to read data from a memory stream using TVPEStream.Read() – where the memory stream has been filled before by WriteDocStream().

Call Seek(0) first!

This applies to all read and write operations.

Example:
Vpe.OpenDoc()

// Write the current document to a memory stream
TVPEStream Stream = Vpe.CreateMemoryStream(0)
Vpe.WriteDocStream(Stream)

// Create a second document
Vpe2.OpenDoc()

// Read the memory stream into the new document, seek to position 0 first!
Stream.Seek(0);
Vpe.ReadDocStream(Stream);

// Cleanup
Stream.Close();

See also:
WriteDocStream
WriteDocStreamPageRange

ReadDocStream
ReadDocStreamPageRange

PictureStream
RenderPictureStream

PictureExportStream
PictureExportPageStream

WriteRTFStream
WriteBoxRTFStream
RenderRTFStream
RenderBoxRTFStream
23.2 CreateMemoryStream

[Professional Edition and above; .NET, Java, PHP, etc.]

Creates a new empty stream in memory. A memory stream can grow as large as you wish, the size is only limited by available memory. VPE divides a memory stream into chunks of equal size - by default 16 KB. This means that an initial stream only occupies 16 KB of memory. If you write more data to a memory stream, the memory stream is enlarged by additional chunks as required. Internally VPE stores each chunk separately and links each chunk with the next.

This is a method of the VPE document. All other methods and properties described throughout this chapter belong to the TVPEStream object, which is created by this method.

method TVPEStream VPE.CreateMemoryStream(
    long chunk_size
)

long chunk_size
    The size of each chunk, if you set this parameter to zero, VPE uses a default chunk size of 16 KB, which is a reasonable value.

Returns:
    A TVPEStream object for the created memory stream.

In case of an error an exception is thrown and LastError is set.

You must ALWAYS close all streams you ever opened or created, when you have done using them. Not doing so will result in memory leaks (only at the moment your application is closed, all streams will be released from memory). Neither calling CloseDoc(), nor destroying a TVPEStream object will close a stream.

Remarks:
    sets LastError
23.3 Close

[Professional Edition and above; .NET, Java, PHP, etc.]

Closes the given stream.

method override void TVPEStream.Close()

Remarks:
When closing a stream, the internal stream handle is destroyed. You may NOT use this object in any further calls to a method, which takes a stream object as parameter.
23.4 Read

[Professional Edition and above; .NET, Java, PHP, etc.]

Reads data from the supplied stream.

```csharp
method override int TVPESStream.Read(
    [InAttribute] [OutAttribute] byte[] buffer,
    int offset,
    int count
)
```

- **byte[] buffer**
  - the buffer, to which the data will be copied from the stream

- **int offset**
  - offset into the buffer

- **int count**
  - the number of bytes that shall be read

**Returns:**
- The number of bytes read from the stream.

**Remarks:**
- When reading from a stream, which was written by a VPE method, e.g. WriteDocStream() or PictureStream() etc., you need to call Seek(0) first before calling Read() in order to reset the internal stream pointer to the beginning of the stream.
23.5 Write

[Professional Edition and above; .NET, Java, PHP, etc.]

Writes data to the supplied stream.

```csharp
method override void TVPEStream.Write(
    [InAttribute] [OutAttribute] byte[] buffer,
    int offset,
    int count
)
```

`byte[] buffer`
- the buffer, from which the data will be copied to the stream

`int offset`
- offset into the buffer

`int count`
- the number of bytes that shall be written

Remarks:
If you want to call a VPE method (for example ReadDocStream() or PictureStream()), after you have written data to a stream, you need to call Seek(0) first before calling Write() in order to reset the internal stream pointer to the beginning of the stream.
23.6 WriteTo

[Professional Edition and above; .NET, Java, PHP, etc.]

Writes the entire contents of this memory stream to another stream.

```c
method virtual void TVPESStream.WriteTo(
    Stream stream
)

Stream stream
    The stream to write this memory stream to.
```
23.7 Length

[Professional Edition and above; .NET, Java, PHP, etc.]

Returns the size - in bytes - of the given stream.

**property override long TVPEStream.Length**

read; runtime only

**Returns:**

The size of the stream in bytes.
23.8 IsEof

[Professional Edition and above; .NET, Java, PHP, etc.]

Returns the End Of File status of the given stream.

This property returns true after the first read operation that attempts to read past the end of the file. IsEof continues to report true, until Seek() (see “Seek (.NET)” on page 611 and “Seek (Java, PHP, etc.)” on page 612) is called.

property virtual boolean TVPEStream.IsEof

read; runtime only

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>EOF</td>
</tr>
<tr>
<td>False</td>
<td>not EOF</td>
</tr>
</tbody>
</table>
23.9 State

[Professional Edition and above; .NET, Java, PHP, etc.]

Returns the status of the stream.

For memory streams the state can become false, if the system is out-of-memory, or if you try to read past the end of file, or if you seek outside the file.

**property boolean TVPEStream.State**

read; runtime only

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>status ok</td>
</tr>
<tr>
<td>False</td>
<td>failure</td>
</tr>
</tbody>
</table>
23.10 Position

[Professional Edition and above; .NET, Java, PHP, etc.]

Returns the current position - in bytes - of the internal file pointer.

**property long TVPESStream.Position**

read; runtime only

**Returns:**

The current position within the stream.
23.11 Seek (.NET)
[Professional Edition and above; .NET only]
Moves the file pointer to the given position.

**method override long TVPESStream.Seek(**
    **long offset,**
    **SeekOrigin loc**
***)

*long offset*
  The new position within the stream. This is relative to the loc parameter, and can be positive or negative.

*SeekOrigin loc*
  A value of type SeekOrigin, which acts as the seek reference point.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin</td>
<td>the beginning of a stream.</td>
</tr>
<tr>
<td>Current</td>
<td>Specifies the current position within a stream.</td>
</tr>
<tr>
<td>End</td>
<td>Specifies the end of a stream.</td>
</tr>
</tbody>
</table>
23.12 Seek (Java, PHP, etc.)

[Professional Edition and above; Java, PHP, etc.]

Moves the file pointer to the given position.

```java
method override long TVPEStream.Seek(
    long pos
)
```

*long pos*

The new position within the stream. This is relative to the beginning of the file.
23.13 SeekEnd (Java, PHP, etc.)

[Professional Edition and above; Java, PHP, etc.]

Moves the file pointer to the given position.

```java
method override long TVPEStream.SeekEnd(
    long pos
)
```

- `long pos`
  The new position within the stream. This is relative to the end of the file.
23.14 SeekRel (Java, PHP, etc.)

[Professional Edition and above; Java, PHP, etc.]

Moves the file pointer to the given position.

```java
method override long TVPESStream.SeekRel(
    long offset
)
```

*long offset*

The offset within the stream, relative to the current position.
23.15 SetLength (.NET)
[Professional Edition and above; .NET only]
Not supported by TVPEStream, throws a NotSupportedException.

method override void TVPEStream.SetLength (  
   long value  
)
23.16 Flush (.NET)

[Professional Edition and above; .NET only]

Overridden, does nothing.

method override void TVPEStream.Flush ( )
23.17 CanRead (.NET)

[Professional Edition and above; .NET only]

Overridden, returns true.

```csharp
property override bool CanRead
    read; runtime only
```
23.18 CanSeek (.NET)
[Professional Edition and above; .NET only]
Overridden, returns true.

property override bool CanSeek

read; runtime only
23.19 CanWrite (.NET)

[Professional Edition and above; .NET only]

Overridden, returns true.

**property override bool CanWrite**

read; runtime only
24 Charts

24.1 Charts

The chart was created with the following short code sequence:
VCL:

```pascal
hData: Pointer
hData = Doc.ChartDataCreate(3, 3) // 3 Rows and 3 Columns
Doc.ChartTitle = "Sales"
Doc.ChartSubTitle = "Northern Region Sales Department, July '98"
Doc.ChartFootNote = "Measuring by Data Measure Inc. 10/98"
Doc.ChartLegendPosition = VCHART_LEGENDPOS_TOP
Doc.ChartDataSetXAxisTitle(hData, "Month")
Doc.ChartDataSetYAxisTitle(hData, "Pieces in thousand")
Doc.ChartDataAddLegend(hData, "200 Mhz CPU's")
Doc.ChartDataAddLegend(hData, "300 Mhz CPU's")
Doc.ChartDataAddLegend(hData, "400 Mhz CPU's")
Doc.ChartDataAddValue(hData, 0, 10) // Row 0
Doc.ChartDataAddValue(hData, 0, 20)
Doc.ChartDataAddValue(hData, 0, 25)
Doc.ChartDataAddValue(hData, 1, 7) // Row 1
Doc.ChartDataAddValue(hData, 1, 10)
Doc.ChartDataAddValue(hData, 1, 20)
Doc.ChartDataAddValue(hData, 2, 3) // Row 2
Doc.ChartDataAddValue(hData, 2, 5)
Doc.ChartDataAddValue(hData, 2, 8)
Doc.ChartXLabelStartValue = 7
Doc.Chart(1, 1, -18, -18, hData, VCHART_3D_BAR)
```

All other component types (ActiveX, .NET, Java, ...):

```pascal
Dim Data as TVPEChartData
Data = Doc.ChartDataCreate(3, 3) // 3 Rows and 3 Columns
Doc.ChartTitle = "Sales"
Doc.ChartSubTitle = "Northern Region Sales Department, July '98"
Doc.ChartFootNote = "Measuring by Data Measure Inc. 10/98"
Doc.ChartLegendPosition = ChartLegendPosition.Top
Data.XAxisTitle = "Month"
Data.YAxisTitle = "Pieces in thousand"
Data.AddLegend("200 Mhz CPU's")
Data.AddLegend("300 Mhz CPU's")
Data.AddLegend("400 Mhz CPU's")
Data.AddValue(0, 10) // Row 0
Data.AddValue(0, 20)
Data.AddValue(0, 25)
Data.AddValue(1, 7) // Row 1
Data.AddValue(1, 10)
Data.AddValue(1, 20)
Data.AddValue(2, 3) // Row 2
Data.AddValue(2, 5)
Data.AddValue(2, 8)
Doc.ChartXLabelStartValue = 7
Doc.Chart(1, 1, -18, -18, Data, ChartType.Bar3D)
```
24.1.1 The SmartChart Technology

The dimensions of the different chart elements - like title, subtitle, etc. - depend on the total size of
the chart. The bigger the dimensions of a chart are, the bigger are the font sizes and line
thicknesses (for line charts). Vice versa the smaller the dimensions of a chart are, the smaller are
the font sizes and line thicknesses.

VPE computes the font- and line sizes optimized depending on the available space. We call that
SmartChart Technology: depending on the items displayed (for example with Title, but without
Subtitle and the Legend positioned on the Left / Top position) the chart algorithm computes the
available space for the visible items and computes font sizes and line thicknesses correspondingly.
VPE offers several properties to specify factors for the font sizes and line thicknesses. With those
factors you can not define absolute font- or pen sizes, but relative font- / pen sizes compared to the
original default values.

24.1.2 In VPE, Charts internally consist of two basic parts

1) The Chart-Data Object

This is the set of numeric values you want to visualize. VPE organizes the data internally in a
table. As with SQL, this table has rows and columns:

<table>
<thead>
<tr>
<th></th>
<th>Column 0</th>
<th>Column 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 0</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>row 1</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>row 2</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>row 3</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>

Note, that rows and columns start with the index 0.

Additionally, the following data related elements are part of the Chart-Data:

- Legend
- Labels of the x- and y-axis
- x- and y-unit signs
- colors and appearance of each data column

For .NET the Chart Data is encapsulated in the TVPEChartData object. For the ActiveX / VCL the
Chart Data object is managed with a handle, i.e. a long integer which identifies the object within
VPE.
2) The Chart-Properties

This is the Title, Subtitle, Footnote and colors for graphical elements like grid and text, etc.

Chart Properties behave like all other properties in VPE. This means you can create charts of different types (e.g. Line Chart, Bar Chart, Pie Chart, etc.) that are all using the same ChartData Object, but each Chart can have its individual properties - for example its own title.

The Chart is - like text, barcodes, etc. - inherited from the Box Object (see "The Object-Oriented Style" in the Programmer's Manual), so the properties for a Box (like BkgMode, PenSize and PenStyle, etc.) apply to the Chart Object.
24.2 ChartDataCreate

Creates an empty ChartData-Object, prepared to hold the numeric data and properties related to
the numeric data. The ChartData object stores the numeric data of a chart. You can create multiple
charts of different Chart-Types (like: Line Chart, Bar Chart, Pie Chart, etc.) out of one ChartData
object. All charts which have assigned the same ChartData object, will display the same ChartData
in a different style.

method TVPEChartData [pointer] VPE.ChartDataCreate(
  long Columns,
  long Rows
)

long Columns
number of columns that shall be reserved

long Rows
number of rows that shall be reserved

Returns:

<table>
<thead>
<tr>
<th>VCL</th>
<th>the handle to the created ChartData object. In case of an error (out of memory) the handle is null (0).</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveX, .NET, Java, ...</td>
<td>a newly created TVPEChartData object reference. This object provides several methods and properties to add data to rows and columns, and to specify the color and appearance of each row. In case of an error (out of memory) the returned object reference is null.</td>
</tr>
</tbody>
</table>

Remarks:
The created ChartData object may only be used within the context of the document where it was created. Don't use a ChartData object in a different document, this will cause a GPF (General Protection Fault).

Example:

VCL:

```
  hData: Pointer
  hData = Doc.ChartDataCreate(2, 4)
```

ActiveX, .NET, Java, ...:

```
  dim Data as TVPEChartData
  Data = Doc.ChartDataCreate(2, 4)
```

Creates a ChartData object with two columns (for example Apples and Bananas), where each column can hold 4 numeric data values ( = rows).
24.3 AddValue

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

Adds a new value to the specified column of the ChartData object.

**method void TVPEChartData.AddValue(  
  int Column,  
  double Value  
)

*int Column*
  
  column, where to add the value

*double Value*
  
  value to add

**Example:**

```vba
Dim Data As TVPEChartData
Data = Doc.ChartDataCreate(2, 4)

Data.AddValue(0, 10)  
Data.AddValue(0, 20)  
Data.AddValue(0, 30)  
Data.AddValue(0, 40)  
Data.AddValue(1, 5)   
Data.AddValue(1, 10)  
Data.AddValue(1, 15)  
Data.AddValue(1, 20)  
```

Creates a ChartData object with two columns (for example Apples and Bananas), where each column can hold 4 numeric data values ( = rows).

Now the internal data table of the Chart Data object will look like this:

<table>
<thead>
<tr>
<th></th>
<th>Column 0</th>
<th>Column 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 0</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>row 1</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>row 2</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>row 3</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>
24.4 AddLegend

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

Adds a new descriptive string for a column to the legend.

```
method void TVPEChartData.AddLegend(
    string Legend
)
```

```
string Legend
legend string
```

Example:
```
dim Data as TVPEChartData
Data = Doc.ChartDataCreate(2, 4)
Data.AddLegend("Apples")
Data.AddLegend("Bananas")
```

Creates a ChartData object with two columns, titled "Apples" and "Bananas" in the legend.
24.5 XAxisTitle

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

Sets the x-axis title of the chart. This is the descriptive text of the x-axis.

property string TVPEChartData.XAxisTitle

write; runtime only

Possible Values:
- title for the x-axis
24.6 YAxisTitle

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

Sets the y-axis title of the chart. This is the descriptive text of the y-axis.

property string TVPEChartData.YAxisTitle

write; runtime only

Possible Values:

title for the y-axis
24.7 AddXLabel

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

Adds a new X-Label to the chart. By default, the x-axis is automatically labeled. With ChartXLabelState() you can switch to user defined labels and then add X-Labels to the chart. (see also ChartXLabelStartValue)

Method void TVPEChartData.AddXLabel(
    string XLabel
)

string XLabel
    string of label to add

Remarks:
You can control if, how and on what position y-labels are drawn with the following properties:

- ChartXLabelState
- ChartXGridStep
- ChartXLabelStep

Example:

```vbnet
dim Data as TVPEChartData
Data = Doc.ChartDataCreate(2, 4)
Data.AddValue(0, 10)
Data.AddValue(0, 20)
Data.AddValue(0, 30)
Data.AddValue(0, 40)
Data.AddValue(1, 5)
Data.AddValue(1, 10)
Data.AddValue(1, 15)
Data.AddValue(1, 20)
Data.AddLegend("Apples")
Data.AddLegend("Bananas")
```

Now the internal data table of the Chart Data object will look like this:

<table>
<thead>
<tr>
<th></th>
<th>Column 0 &quot;Apples&quot;</th>
<th>Column 1 &quot;Bananas&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 0</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>row 1</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>row 2</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>row 3</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>
With the following code:

```csharp
Doc.ChartXLabelState = ChartLabelState.User
Data.AddXLabel("1. Quarter")
Data.AddXLabel("2. Quarter")
Data.AddXLabel("3. Quarter")
Data.AddXLabel("4. Quarter")
```

the internal data table of the Chart Data object will look like this:

<table>
<thead>
<tr>
<th></th>
<th>Column 0 &quot;Apples&quot;</th>
<th>Column 1 &quot;Bananas&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 0</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>row 1</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>row 2</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>row 3</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>
24.8 AddYLabel

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

Adds a new Y-Label to the chart. By default, the y-axis is labeled automatically. With ChartYLabelState you can switch to user defined labels and then add Y-Labels to the chart. Each label that would automatically be drawn can be replaced with this method.

method void TVPEChartData.AddYLabel(
    string YLabel
)

string YLabel
    string of label to add

Remarks:
    You can control if, how and on what position y-labels are drawn with the following properties:
    • ChartYLabelState
    • ChartYGridStep
    • ChartYLabelStep
    • ChartYLabelDivisor
    • ChartDataSetMinimum
    • ChartDataSetMaximum
24.9 SetColor

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

By default, VPE assigns itself default colors to each data column. With this method you can specify individual colors for each column.

```
method void TVPEChartData.SetColor(
    int Column,
    Color Color
)
```

- `int Column` column the color shall be assigned to
- `Color Color` any of the "COLOR_xyz" constants described in Programmer's Manual or any member of the .NET Color structure
24.10 SetLineStyle

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

With this method you can specify individual line styles for each column.

method void TVPEChartData.SetLineStyle(
    int Column,
    PenStyle LineStyle
)

*int Column*
- column the line style shall be assigned to

*PenStyle [integer] LineStyle*
- possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solid</td>
<td></td>
</tr>
<tr>
<td>Dash</td>
<td>------</td>
</tr>
<tr>
<td>Dot</td>
<td>......</td>
</tr>
<tr>
<td>DashDot</td>
<td><em>.</em>.<em>.</em></td>
</tr>
<tr>
<td>DashDotDot</td>
<td><em>..</em>.._</td>
</tr>
</tbody>
</table>

**Default:**
- PenStyle.Solid

**Example:**

```plaintext
Data.SetLineStyle(0, PenStyle.Solid)
```
24.11 SetHatchStyle

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

If you create a Bar-, Area- or Pie Chart from the Chart Data, you can specify individual hatch styles for each column with this method. By default, no hatching is used.

**method void TVPEChartData.SetHatchStyle(**

- **int Column,**
- **HatchStyle HatchStyle**

**)

*int Column*

column the line style shall be assigned to

*HatchStyle HatchStyle*

possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>see HatchStyle – possible styles</td>
</tr>
<tr>
<td>Horizontal</td>
<td>see HatchStyle – possible styles</td>
</tr>
<tr>
<td>Vertical</td>
<td>see HatchStyle – possible styles</td>
</tr>
<tr>
<td>ForwardDiagonal</td>
<td>see HatchStyle – possible styles</td>
</tr>
<tr>
<td>BackwardDiagonal</td>
<td>see HatchStyle – possible styles</td>
</tr>
<tr>
<td>Cross</td>
<td>see HatchStyle – possible styles</td>
</tr>
<tr>
<td>DiagonalCross</td>
<td>see HatchStyle – possible styles</td>
</tr>
</tbody>
</table>

**Default:**

HatchStyle.None (no hatching is used)

**Example:**

```csharp
Data.SetHatchStyle(0, HatchStyle.DiagonalCross)
```
24.12 SetPointType
[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

If you create a Point Chart from the Chart Data, you can specify individual point styles for each column with this method.

**method void TVPEChartData.SetPointType(**
    int Column,
    ChartSymbol PointType
**)

*int Column*
   column the color shall be assigned to

*ChartSymbol PointType*
possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Square</td>
<td></td>
</tr>
<tr>
<td>Triangle</td>
<td></td>
</tr>
<tr>
<td>Circle</td>
<td></td>
</tr>
<tr>
<td>Cross</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Point</td>
<td></td>
</tr>
</tbody>
</table>

**Default:**
ChartSymbol.X

**Example:**
Data.SetPointType(0, ChartSymbol.Square)
24.13 Minimum
[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

VPE determines the minimum value of the numeric data stored in a Chart Data object itself. You can change the minimum value with this method.

**property double TVPEChartData.Minimum**

write; runtime only

**Possible Values:**
the minimum value of the numeric data of the Chart Data object

**Remarks:**
Because VPE compares - and maybe updates - the current determined minimum value with each value added by TVPEChartData.AddValue(), you should call this method *after* you have finished adding values.
24.14 Maximum

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

VPE determines the maximum value of the numeric data stored in a Chart Data object itself. You can change the maximum value with this method.

**property double TVPEChartData.Maximum**

write; runtime only

**Possible Values:**

the maximum value of the numeric data of the Chart Data object

**Remarks:**

Because VPE compares - and maybe updates - the current determined [minimum value](#) with each value added by TVPEChartData.AddValue(), you should call this method after you have finished adding values.
24.15 AddGap

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

Adds a gap to the data. This is especially useful for line charts in case a data value is missing. If you would add a zero, the line would be drawn to the zero-value, but this wouldn’t reflect, that there is NO value.

```csharp
method void TVPEChartData.AddGap(
    int Column
)

int Column
    the column where to add the gap
```
24.16 AddRow

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

Resizes the internal data table of the ChartData object, so it can hold one more new row per column. This is especially useful if you don’t know in advance, how many rows the ChartData data table will have and you need to add rows while querying your data source (or database).

method void TVPEChartData.AddRow()
24.17 AddColumn

[ActiveX, .NET, Java, PHP, etc., method of the TVPEChartData object]

Resizes the internal data table of the ChartData object, and adds a new empty column. This is especially useful if you don’t know in advance, how many columns the ChartData data table will have and you need to add columns while querying your data source (or database).

method void TVPEChartData.AddColumn( )
24.18 ChartDataAddValue

[VCL only]

Adds a new value to the specified column of the ChartData object specified in the parameter hData.

**method** void VPE.ChartDataAddValue(
    pointer hData,
    long Column,
    double Value
)

*pointer* hData
    handle to ChartData object

*long* Column
    column, where to add the value

*double* Value
    value to add

**Example:**

```plaintext
hData: Pointer
hData = Doc.ChartDataCreate(2, 4)

Creates a ChartData object with two columns (for example Apples and Bananas), where each column can hold 4 numeric data values ( = rows).
```

```plaintext
Doc.ChartDataAddValue(hData, 0, 10)
Doc.ChartDataAddValue(hData, 0, 20)
Doc.ChartDataAddValue(hData, 0, 30)
Doc.ChartDataAddValue(hData, 0, 40)

Doc.ChartDataAddValue(hData, 1, 5)
Doc.ChartDataAddValue(hData, 1, 10)
Doc.ChartDataAddValue(hData, 1, 15)
Doc.ChartDataAddValue(hData, 1, 20)
```

Now the internal data table of the Chart Data object will look like this:

<table>
<thead>
<tr>
<th></th>
<th>Column 0</th>
<th>Column 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 0</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>row 1</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>row 2</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>row 3</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>
24.19 ChartDataAddLegend

[VCL only]
Adds a new descriptive string for a column to the legend.

**method void VPE.ChartDataAddLegend(**
pointer hData,
string Legend
**)

*pointer hData*
handle to ChartData object

*string Legend*
legend string

**Example:**
```
hData: Pointer
hData = Doc.ChartDataCreate(2, 4)
Doc.ChartDataAddLegend(hData, "Apples")
Doc.ChartDataAddLegend(hData, "Bananas")
```

Creates a ChartData object with two columns, titled "Apples" and "Bananas" in the legend.
24.20 ChartDataSetXAxisTitle

[VCL only]
Sets the x-axis title of the chart. This is the descriptive text of the x-axis.

```java
method void VPE.ChartDataSetXAxisTitle(
    pointer hData,
    string XAxisTitle
)
```

`pointer hData`
handle to ChartData object

`string XAxisTitle`
title for x-axis
24.21 ChartDataSetYAxisTitle

[VCL only]
Sets the y-axis title of the chart. This is the descriptive text of the y-axis.

**method void VPE.ChartDataSetYAxisTitle(**
  **    pointer hData,**
  **    string YAxisTitle**

  **)**

*pointer hData*
  handle to ChartData object

*string YAxisTitle*
  title for y-axis
24.22 ChartDataAddXLabel

[VCL only]

Adds a new X-Label to the chart. By default, the x-axis is automatically labeled. With ChartXLabelState you can switch to user defined labels and then add X-Labels to the chart. (see also ChartXLabelStartValue)

method void VPE.ChartDataAddXLabel(
    pointer hData,  
    string XLabel  
)

pointer hData
    handle to ChartData object

string XLabel
    string of label to add

Remarks:
You can control if, how and on what position y-labels are drawn with the following properties:

- ChartXLabelState
- ChartXGridStep
- ChartXLabelStep

Example:

hData: Pointer  
hData = Doc.ChartDataCreate(2, 4)  

Creates a ChartData object with two columns (for example Apples and Bananas), where each column can hold 4 numeric data values (= rows).

```
Doc.ChartDataAddValue(hData, 0, 10)
Doc.ChartDataAddValue(hData, 0, 20)
Doc.ChartDataAddValue(hData, 0, 30)
Doc.ChartDataAddValue(hData, 0, 40)
Doc.ChartDataAddValue(hData, 1, 5)
Doc.ChartDataAddValue(hData, 1, 10)
Doc.ChartDataAddValue(hData, 1, 15)
Doc.ChartDataAddValue(hData, 1, 20)
Doc.ChartDataAddLegend(hData, "Apples")
Doc.ChartDataAddLegend(hData, "Bananas")
```

Now the internal data table of the Chart Data object will look like this:

<table>
<thead>
<tr>
<th></th>
<th>Column 0 &quot;Apples&quot;</th>
<th>Column 1 &quot;Bananas&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 0</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>row 1</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>row 2</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>row 3</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>
With the following code:

```vbnet
Doc.ChartXLabelState = VCHART_LABEL_USER
Doc.ChartDataAddXLabel(hData, "1. Quarter")
Doc.ChartDataAddXLabel(hData, "2. Quarter")
Doc.ChartDataAddXLabel(hData, "3. Quarter")
Doc.ChartDataAddXLabel(hData, "4. Quarter")
```

the internal data table of the Chart Data object will look like this:

<table>
<thead>
<tr>
<th></th>
<th>Column 0 &quot;Apples&quot;</th>
<th>Column 1 &quot;Bananas&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 0</td>
<td>&quot;1. Quarter&quot;</td>
<td>10</td>
</tr>
<tr>
<td>row 1</td>
<td>&quot;2. Quarter&quot;</td>
<td>20</td>
</tr>
<tr>
<td>row 2</td>
<td>&quot;3. Quarter&quot;</td>
<td>30</td>
</tr>
<tr>
<td>row 3</td>
<td>&quot;4. Quarter&quot;</td>
<td>40</td>
</tr>
</tbody>
</table>
24.23 ChartDataAddYLabel

[VCL only]

Adds a new Y-Label to the chart. By default, the y-axis is labeled automatically. With ChartYLabelState you can switch to user defined labels and then add Y-Labels to the chart. Each label that would automatically be drawn can be replaced with this method.

```cpp
method void VPE.ChartDataAddYLabel(
    pointer hData,
    string YLabel
)
```

*pointer hData*  
handle to ChartData object

*string YLabel*  
string of label to add

**Remarks:**
You can control if, how and on what position y-labels are drawn with the following properties:

- ChartYLabelState
- ChartYGridStep
- ChartYLabelStep
- ChartYLabelDivisor
- ChartDataSetMinimum
- ChartDataSetMaximum
24.24 ChartDataSetColor

[VCL only]
By default, VPE assigns itself default colors to each data column. With this method you can specify individual colors for each column.

**method void VPE.ChartDataSetColor(**
    pointer hData,
    long Column,
    Color Color
)

*pointer hData*
    handle to **ChartData object**

*long Column*
    column the color shall be assigned to

*Color Color*
    any of the "COLOR_xyz" constants described in Programmer's Manual
    or any RGB value
24.25 ChartDataSetLineStyle
[VCL only]
With this method you can specify individual line styles for each column.

method void VPE.ChartDataSetLineStyle(
    pointer hData,
    long Column,
    PenStyle [integer] LineStyle
)

pointer hData
    handle to ChartData object

long Column
    column the line style shall be assigned to

PenStyle [integer] LineStyle
    possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>psSolid</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>psDash</td>
<td>1</td>
<td>-------</td>
</tr>
<tr>
<td>psDot</td>
<td>2</td>
<td>..........</td>
</tr>
<tr>
<td>psDashDot</td>
<td>3</td>
<td><em>.</em>.<em>.</em>.</td>
</tr>
<tr>
<td>psDashDotDot</td>
<td>4</td>
<td><em>.</em>.<em>.</em>.</td>
</tr>
</tbody>
</table>

Default:
    psSolid

Example:
    Doc.ChartDataSetLineStyle(hData, 0, psSolid)
24.26 ChartDataSetHatchStyle

[VCL only]

If you create a Bar-, Area- or Pie Chart from the Chart Data, you can specify individual hatch styles for each column with this method. By default, no hatching is used.

method void VPE.ChartDataSetHatchStyle(
    pointer hData,
    long Column,
    integer HatchStyle
)

pointer hData
    handle to ChartData object

long Column
    column the line style shall be assigned to

integer HatchStyle
    column the hatch style shall be assigned to; possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>hsNone</td>
<td>-1</td>
<td>see HatchStyle – possible styles</td>
</tr>
<tr>
<td>hsHorizontal</td>
<td>0</td>
<td>see HatchStyle – possible styles</td>
</tr>
<tr>
<td>hsVertical</td>
<td>1</td>
<td>see HatchStyle – possible styles</td>
</tr>
<tr>
<td>hsFDiagonal</td>
<td>2</td>
<td>see HatchStyle – possible styles</td>
</tr>
<tr>
<td>hsBDiagonal</td>
<td>3</td>
<td>see HatchStyle – possible styles</td>
</tr>
<tr>
<td>hsCross</td>
<td>4</td>
<td>see HatchStyle – possible styles</td>
</tr>
<tr>
<td>hsDiagCross</td>
<td>5</td>
<td>see HatchStyle – possible styles</td>
</tr>
</tbody>
</table>

Default:
    hsNone (no hatching is used)

Example:
    Doc.ChartDataSetHatchStyle(hData, 0, hsDiagCross)
24.27 ChartDataSetPointType

[VCL only]

If you create a Point Chart from the Chart Data, you can specify individual point styles for each column with this method.

**method void VPE.ChartDataSetPointType(**
  pointer hData,
  long Column,
  ChartSymbol [long] PointType
**)

*pointer hData*
  handle to [ChartData](#) object

*long Column*
  column the color shall be assigned to

*ChartSymbol [long] PointType*
  possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCHART_SYMBOL_NONE</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>VCHART_SYMBOL_SQUARE</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VCHART_SYMBOL_TRIANGLE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>VCHART_SYMBOL_CIRCLE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>VCHART_SYMBOL_CROSS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>VCHART_SYMBOL_X</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>VCHART_SYMBOL_POINT</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

**Default:**
VCHART_SYMBOL_X

**Example:**

```
Doc.ChartDataSetPointType(hData, 0, VCHART_SYMBOL_SQUARE)
```
24.28 ChartDataSetMinimum

[VCL only]

VPE determines the minimum value of the numeric data stored in a Chart Data object itself. You can change the minimum value with this method.

**method** void VPE.ChartDataSetMinimum(
    pointer hData,
    double Minimum
)

*pointer hData*
handle to ChartData object

*double Minimum*
the minimum value of the numeric data in a Chart Data object

**Remarks:**
Because VPE compares - and maybe updates - the current determined minimum value with each value added by ChartDataAddValue(), you should call this method after you have finished adding values.
24.29 ChartDataSetMaximum

[VCL only]

VPE determines the maximum value of the numeric data stored in a Chart Data object itself. You can change the maximum value with this method.

```c
method void VPE.ChartDataSetMaximum(
    pointer hData,
    double Maximum
)
```

- `pointer hData`
  handle to ChartData object

- `double Maximum`
  the maximum value of the numeric data in a Chart Data object

Remarks:
Because VPE compares - and maybe updates - the current determined minimum value with each value added by ChartDataAddValue(), you should call this method **after** you have finished adding values.
24.30 ChartDataAddGap

[VCL only]

Adds a gap to the data. This is especially useful for line charts in case a data value is missing. If you would add a zero, the line would be drawn to the zero-value, but this wouldn’t reflect, that there is NO value.

```c
method void VPE.ChartDataAddGap(
    pointer hData,
    long Column
)
```

`pointer hData`
- handle to ChartData object

`long Column`
- the column where to add the gap
24.31 ChartDataAddRow

[VCL only]
Resizes the internal data table of the ChartData object specified in the Parameter hData, so it can hold one more new row per column. This is especially useful if you don’t know in advance, how many rows the ChartData data table will have and you need to add rows while querying your data source (or database).

method void VPE.ChartDataAddRow(
    pointer hData
)

pointer hData
handle to ChartData object
24.32 ChartDataAddColumn

[VCL only]

Resizes the internal data table of the ChartData object specified in the Parameter hData, and adds a new empty column. This is especially useful if you don’t know in advance, how many columns the ChartData data table will have and you need to add columns while querying your data source (or database).

```plaintext
method void VPE.ChartDataAddColumn(
    pointer hData
)

pointer hData
    handle to ChartData object
```
24.33 ChartTitle

(Chart Property) Sets the title property for the next created chart object.

property string VPE.ChartTitle

    write; runtime only

Possible Values:
    title of the chart
24.34 ChartTitleFontName

(Chart Property) Sets the font for the title of the next created chart object.

property string VPE.ChartTitleFontName

  write; runtime only

Possible Values:
  name of the font

Default:
  Times New Roman (Times on non-Windows platforms)
24.35 ChartTitleFontSizeFactor

(Chart Property) Specifies a factor for the font size of the chart title.

property double VPE.ChartTitleFontSizeFactor

    write; runtime only

Possible Values:
    0.1 <= factor <= 10

Default:
    1.0
24.36 ChartSubTitle

(Chart Property) Sets the subtitle property for the next created chart object.

**property string VPE.ChartSubTitle**

write; runtime only

**Possible Values:**
subtitle of the chart
24.37 ChartSubTitleFontSizeFactor

(Chart Property) Specifies a factor for the font size of the chart's subtitle.

property double VPE.ChartSubTitleFontSizeFactor

    write; runtime only

Possible Values:
    0.1 <= factor <= 10

Default:
    0.6
24.38 ChartFootNote
(Chart Property) Sets the footnote property for the next created chart object.

property string VPE.ChartFootNote
  write; runtime only

Possible Values:
  footnote of the chart
24.39 ChartFootNoteFontName

(Chart Property) Sets the font of the footnote for the next created chart object.

**property string VPE.ChartFootNoteFontName**

- write; runtime only

**Possible Values:**
- name of the font

**Default:**
- Times New Roman (Times on non-Windows platforms)
24.40 ChartFootNoteFontSizeFactor
(Chart Property) Specifies a factor for the font size of the chart's footnote.

property double VPE.ChartFootNoteFontSizeFactor
   write; runtime only

Possible Values:
   0.1 <= factor <= 10

Default:
   0.5
24.41 ChartAxesFontName

(Chart Property) Specifies the font name of the chart's x- and y-axis titles.

property string VPE.ChartAxesFontName

write; runtime only

Possible Values:
name of the font

Default:
Times New Roman (Times on non-Windows platforms)
24.42 ChartAxisTitleFontSizeFactor

(Chart Property) Specifies a factor for the font size of the chart's x- and y-axis titles.

property double VPE.ChartAxisTitleFontSizeFactor

    write; runtime only

Possible Values:

    0.1 <= factor <= 3

Default:

    1.0
24.43 ChartLegendFontName

(Chart Property) Sets the font of the legend for the next created chart object.

property string VPE.ChartLegendFontName

   write; runtime only

Possible Values:
   name of the font

Default:
   Times New Roman (Times on non-Windows platforms)
24.44 ChartLegendFontSizeFactor
(Chart Property) Specifies a factor for the font size of the chart's legend.

property double VPE.ChartLegendFontSizeFactor
    write; runtime only

Possible Values:
    possible values 0.1 <= factor <= 3

Default:
    1.4
24.45 ChartLineWidthFactor

(Chart Property) Line Charts only: Specifies a factor for the line thickness.

property double VPE.ChartLineWidthFactor

   write; runtime only

Possible Values:
   possible values 0.1 <= factor <= 8

Default:
   3.0
24.46 ChartBarWidthFactor

(Chart Property) Bar Charts only (2D and 3D): Specifies a factor for bar width.

property double VPE.ChartBarWidthFactor

write; runtime only

Possible Values:
possible values 0.2 <= factor <= 1.9

Default:
1.0
24.47 ChartRow

(Chart Property) For Pie Chart only. Selects the row of the Chart Data object which shall be used by a pie chart. A Pie Chart can only visualize one row of data.

property long VPE.ChartRow

   write; runtime only

Possible Values:
   the row that shall be visualized by a Pie Chart

Default:
   0 ( = the pie chart is created out of the first row of the Chart Data object)

Example:

   ActiveX / VCL:
   Doc.ChartRow = 0
   Doc.Chart(1, 1, -18, -18, hData, VCHART_3D_PIE)

   .NET:
   Doc.ChartRow = 0
   Doc.Chart(1, 1, -18, -18, Data, ChartType.Pie3D)

   Creates a 3D Pie Chart from the 1st row of a Chart Data object.

   ActiveX / VCL:
   Doc.ChartRow = 1
   Doc.Chart(1, 1, -18, -18, hData, VCHART_PIE)

   .NET:
   Doc.ChartRow = 1
   Doc.Chart(1, 1, -18, -18, Data, ChartType.Pie)

   Creates a 2D Pie Chart from the 2nd row of a Chart Data object.
24.48 ChartGridBkgColor

(Chart Property) Sets the chart grid background color for the next created chart object.

**property Color VPE.ChartGridBkgColor**

 - write; runtime only

**Possible Values:**
- any of the “COLOR_xyz” constants described in Programmer's Manual
- or ActiveX / VCL: any RGB value
- or .NET: any member of the .NET Color structure

**Default:**
- COLOR_BLACK
24.49 ChartGridBkgMode

(Chart Property) Sets the chart grid background mode for the next created chart object.

property BkgMode [integer] VPE.ChartGridBkgMode

write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBKG_SOLID</td>
<td>0</td>
<td>Solid</td>
<td>solid background color</td>
</tr>
<tr>
<td>VBKG_TRANSPARENT</td>
<td>1</td>
<td>Transparent</td>
<td>transparent background</td>
</tr>
</tbody>
</table>

Default:

VBKG_TRANSPARENT for 2-D charts and VBKG_SOLID for 3-D charts.

Remarks:
The chart grid background is the area within the grid. Using other values from the BkgMode enumeration has no effect., because the background mode for the whole chart object is set with BkgMode.
24.50 ChartGridType

(Chart Property) Sets the grid type for the next created chart object.

property ChartGridType [integer] VPE.ChartGridType

write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCHART_GRID_NONE</td>
<td>-1</td>
<td>None</td>
<td>No Grid is drawn</td>
</tr>
<tr>
<td>VCHART_GRID_BOTH_AXIS</td>
<td>0</td>
<td>BothAxis</td>
<td>Grid is drawn for both axis</td>
</tr>
<tr>
<td>VCHART_GRID_X_AXIS</td>
<td>1</td>
<td>XAxis</td>
<td>Grid is only drawn for the x-axis</td>
</tr>
<tr>
<td>VCHART_GRID_Y_AXIS</td>
<td>2</td>
<td>YAxis</td>
<td>Grid is only drawn for the y-axis</td>
</tr>
</tbody>
</table>

Default: VCHART_GRID_BOTH_AXIS
24.51 ChartGridColor

(Chart Property) Sets the grid color for the next created chart object.

**property Color VPE.ChartGridColor**

write; runtime only

**Possible Values:**
- any of the "COLOR_xyz" constants described in Programmer's Manual
- or **ActiveX / VCL**: any RGB value
- or **.NET**: any member of the .NET Color structure
24.52 ChartXGridStep

(Chart Property) Sets the x-grid-step for the next created chart object.

The x-grid-step determines at what position on the x-axis an x-grid-line is drawn. In addition it controls at what position a marker is drawn at the x-axis.

property int VPE.ChartXGridStep

   write; runtime only

Possible Values:
   the grid stepping

Default:
   1

Example:

   Doc.ChartXGridStep = 2

   means, an x-grid line and a marker are drawn on every second position
24.53 ChartYGridStep

(Chart Property) Sets the y-grid-step for the next created chart object. The y-grid-step determines at what value on the y-axis a y-grid-line is drawn.

property double VPE.ChartYGridStep

write; runtime only

Possible Values:
the grid stepping

Default:
Auto

Example:

Doc.ChartYGridStep = 5
means, a y-grid line is drawn at y-values in a distance of 5 (e.g. a y-grid line is drawn at -10, -5, 0, 5, 10, 15, 20, ...)

Doc.ChartYGridStep = 10
means, a y-grid line is drawn at y-values in a distance of 10 (e.g. a y-grid line is drawn at -20, -10, 0, 10, 20, 30, ...)
24.54 SetChartYAutoGridStep

(Chart Property) Instructs VPE to determine itself the grid stepping to gain readable results depending on the scale of the chart.

```java
method void VPE.SetChartYAutoGridStep()
```

**Remarks:**
This is the default behavior of VPE. You can turn the auto stepping off by setting `ChartYGridStep` to any value.
24.55 ChartLegendPosition

(Chart Property) Sets the legend position for the next created chart object.

property ChartLegendPosition [long] VPE.ChartLegendPosition

write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCHART_LEGENDPOS_NONE</td>
<td>-1</td>
<td>None</td>
<td>No legend drawn</td>
</tr>
<tr>
<td>VCHART_LEGENDPOS_RIGHT</td>
<td>0</td>
<td>Right</td>
<td></td>
</tr>
<tr>
<td>VCHART_LEGENDPOS_RIGHT_TOP</td>
<td>1</td>
<td>RightTop</td>
<td></td>
</tr>
<tr>
<td>VCHART_LEGENDPOS_RIGHT_BOTTOM</td>
<td>2</td>
<td>RightBottom</td>
<td></td>
</tr>
<tr>
<td>VCHART_LEGENDPOS_LEFT</td>
<td>3</td>
<td>Left</td>
<td></td>
</tr>
<tr>
<td>VCHART_LEGENDPOS_LEFT_TOP</td>
<td>4</td>
<td>LeftTop</td>
<td></td>
</tr>
<tr>
<td>VCHART_LEGENDPOS_LEFT_BOTTOM</td>
<td>5</td>
<td>LeftBottom</td>
<td></td>
</tr>
<tr>
<td>VCHART_LEGENDPOS_TOP</td>
<td>6</td>
<td>Top</td>
<td></td>
</tr>
<tr>
<td>VCHART_LEGENDPOS_BOTTOM</td>
<td>7</td>
<td>Bottom</td>
<td></td>
</tr>
</tbody>
</table>

Default:
VCHART_LEGENDPOS_LEFT_TOP
24.56 ChartLegendBorderStat

(Chart Property) Determines, whether the legend is drawn with a frame and a shadow or not for the next created chart object.

property boolean [long] VPE.ChartLegendBorderStat

    write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>drawn</td>
</tr>
<tr>
<td>False</td>
<td>invisible</td>
</tr>
</tbody>
</table>

Default:

True
24.57 ChartXLabelState

(Chart Property) Determines, how the x-axis labels are drawn for the next created chart object.

property ChartLabelState [integer] VPE.ChartXLabelState

write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCHART_LABEL_NONE</td>
<td>-1</td>
<td>None</td>
<td>No labels are drawn</td>
</tr>
<tr>
<td>VCHART_LABEL_USER</td>
<td>0</td>
<td>User</td>
<td>User defined labels are drawn</td>
</tr>
<tr>
<td>VCHART_LABEL_AUTO</td>
<td>1</td>
<td>Auto</td>
<td>Labeling is done automatically</td>
</tr>
</tbody>
</table>

Default:

VCHART_LABEL_AUTO ( = labeling is done automatically)
24.58 ChartPieLegendWithPercent

(Chart Property) Pie Chart only: Determines, whether the percent values of each pie element are shown in the legend.

property boolean VPE.ChartPieLegendWithPercent

   write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, the percent values are shown</td>
</tr>
<tr>
<td>False</td>
<td>no, they are not shown</td>
</tr>
</tbody>
</table>

Default:

   True: yes, the percent values are shown
24.59 ChartPieLabelType

(Chart Property) Pie Chart only: Determines, how the labels of a pie chart are drawn.

property ChartPieLabelType [integer] VPE.ChartPieLabelType

write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCHART PIE LABEL NONE</td>
<td>0</td>
<td>None</td>
<td>no labels are drawn</td>
</tr>
<tr>
<td>VCHART PIE LABEL PERCENTAGE</td>
<td>1</td>
<td>Percentage</td>
<td>the percentage values are drawn</td>
</tr>
<tr>
<td>VCHART PIE LABEL LEGEND</td>
<td>2</td>
<td>Legend</td>
<td>the legend texts are used for the labels</td>
</tr>
<tr>
<td>VCHART PIE LABEL XLABELS</td>
<td>3</td>
<td>Xlabels</td>
<td>the user-defined x-labels are used</td>
</tr>
</tbody>
</table>

Default:

PIE_LABEL_PERCENTAGE
24.60 ChartXLabelFontSizeFactor
(Chart Property) Specifies a factor for the font size of the x-axis labels.

property double VPE.ChartXLabelFontSizeFactor

write; runtime only

Possible Values:
0.1 <= factor <= 3

Default:
1.5
24.61 ChartXLabelStep

(Chart Property) Sets the x-label-step for the next created chart object. The x-label-step determines the n-th x-grid-line where a label is drawn.

property long VPE.ChartXLabelStep

write; runtime only

Possible Values:

the n-th x-grid line where a label is drawn

Default:

1 ( = each x-grid line a label is drawn)

Remarks:

This property is always in effect, regardless if ChartXLabelState is VCHART_LABEL_USER or VCHART_LABEL_AUTO.

Example:

```
Doc.ChartXLabelStep = 1
```

means, at each x-grid line a label is drawn at the x-axis.

```
Doc.ChartXLabelStep = 2
```

means, that a label is only drawn each 2nd x-grid line.
24.62 ChartXLabelAngle

(Chart Property) Sets the drawing angle (clockwise) for the x-axis labels.

property integer VPE.ChartXLabelAngle

    write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 degrees, labels are drawn horizontally</td>
</tr>
<tr>
<td>900</td>
<td>90 degrees, labels are drawn vertically</td>
</tr>
<tr>
<td>1800</td>
<td>180 degrees, labels are drawn horizontally</td>
</tr>
<tr>
<td>2700</td>
<td>270 degrees, labels are drawn vertically</td>
</tr>
</tbody>
</table>

Default:

    0
24.63 ChartXLabelStartValue

(Chart Property) Sets the start value for the x-labels. If the x-labeling is done automatically by
VPE (i.e. ChartXLabelState = VCHART_LABEL_AUTO), this is the start value where VPE will
start to number each row.

property long VPE.ChartXLabelStartValue

    write; runtime only

Possible Values:
    the start value for x-labels

Default:
    0 (= labeling will start at 0)

Example:
    By default, VPE will label the x-axis with "0, 1, 2, 3, ..."

    with:

        Doc.ChartXLabelStartValue = 7

    VPE will label the x-axis with "7, 8, 9, 10, ..."
24.64 ChartYLabelState
(Chart Property) Determines, how the y-axis labels are drawn for the next created chart object.

property ChartYLabelState [integer] VPE.ChartYLabelState

  write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCHART_LABEL_NONE</td>
<td>-1</td>
<td>None</td>
<td>No labels are drawn</td>
</tr>
<tr>
<td>VCHART_LABEL_USER</td>
<td>0</td>
<td>User</td>
<td>User defined labels are drawn</td>
</tr>
<tr>
<td>VCHART_LABEL_AUTO</td>
<td>1</td>
<td>Auto</td>
<td>Labeling is done automatically</td>
</tr>
</tbody>
</table>

Default:
VCHART_LABEL_AUTO
24.65 ChartYLabelFontSizeFactor

(Chart Property) Specifies a factor for the font size of the y-axis labels.

property double VPE.ChartYLabelFontSizeFactor

  write; runtime only

Possible Values:
  0.1 <= factor <= 3

Default:
  1.5
24.66 ChartLabelsFontName

(Chart Property) Specifies the font name of the chart's x- and y-axis labels.

property string VPE.ChartLabelsFontName

write; runtime only

Possible Values:
name of the font

Default:
Arial (Helvetica on non-Windows platforms)
24.67 ChartYLabelStep

(Chart Property) Sets the y-label-step for the next created chart object. The y-label-step determines the n-th y-grid-line where a label is drawn.

property long VPE.ChartYLabelStep

  write; runtime only

Possible Values:
  the n-th y-grid line where a label is drawn

Default:
  1 ( = each y-grid line a label is drawn)

Remarks:
  This property is always in effect, regardless if ChartYLabelState is VCHART_LABEL_USER or VCHART_LABEL_AUTO.

Example:

  Doc.ChartYLabelStep = 1
  means, at each y-grid line a label is drawn at the y-axis.

  Doc.ChartYLabelStep = 2
  means, that a label is only drawn each 2nd y-grid line.
24.68 ChartYLabelDivisor

(Chart Property) Sets the divisor for y-labels, if the y-labels are drawn automatically by VPE (ChartYLabelState = VCHART_LABEL_AUTO).

property double VPE.ChartYLabelDivisor

   write; runtime only

Possible Values:
   the divisor for automatically drawn y-labels

Default:
   1

Example:
   Imagine, the ChartData object consists of one column with the values 1000, 2000, 3000
   for ChartYLabelDivisor = 1 the y-labels are drawn as: "1000, 2000, 3000"
   for ChartYLabelDivisor = 1000 the y-labels are drawn as: "1, 2, 3"
24.69 ChartGridRotation

(Chart Property) Sets the grid rotation (clockwise) for the next created chart object. The chart grid (i.e. the chart itself, not the title or legend, etc.) can be rotated by 90 degrees to the right only. This property has no effect on Pie Charts.

**property integer VPE.ChartGridRotation**

write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 degrees</td>
</tr>
<tr>
<td>900</td>
<td>90 degrees, the chart is rotated by 90 degrees clockwise</td>
</tr>
</tbody>
</table>

Default: 0
24.70 ChartYAxisAngle

(Chart Property) Only 3D charts. Sets the view angle of the y-axis. Values are in 1/10°.

**property integer VPE.ChartYAxisAngle**

write; runtime only

**Possible Values:**

150 <= yangle <= 780

**Default:**

300 ( = 30 degrees)

**Remarks:**

With this property you can change the view perspective, it does not rotate the chart itself.
24.71 ChartXAxisAngle

(Chart Property) Only 3D charts. Sets the view angle of the x-axis. Values are in 1/10°.

property integer VPE.ChartXAxisAngle

write; runtime only

Possible Values:
150 <= xangle <= 780

Default:
450 ( = 45 degrees)

Remarks:
With this property you can change the view perspective, it does not rotate the chart itself.
24.72 Chart

Creates a chart object at the given position from the given ChartData object. The type of the chart is specified in the parameter ChartType. All current chart property settings apply to the created chart.

The property TextColor - which is the generic foreground color - specifies the color for the following chart elements:

- title, subtitle, footnote
- x- / y-axis titles
- x- / y-labels

**method void VPE.Chart(****
  VpeCoord Left,
  VpeCoord Top,
  VpeCoord Right,
  VpeCoord Bottom,
  TVPEChartData [pointer] hData,
  ChartType [integer] ChartType
)

**VpeCoord Left, Top, Right, Bottom**
position and dimensions of the chart

**TVPEChartData [pointer] hData**
Chart Data object

**ChartType [integer] ChartType**
possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCHART_POINT</td>
<td>0</td>
<td>Point</td>
</tr>
<tr>
<td>VCHART_LINE</td>
<td>1</td>
<td>Line</td>
</tr>
<tr>
<td>VCHART_BAR</td>
<td>2</td>
<td>Bar</td>
</tr>
<tr>
<td>VCHART_STACKED_BAR_ABSOLUTE</td>
<td>3</td>
<td>StackedBarAbsolute</td>
</tr>
<tr>
<td>VCHART_STACKED_BAR_PERCENT</td>
<td>4</td>
<td>StackedBarPercent</td>
</tr>
<tr>
<td>VCHART_3D_BAR</td>
<td>5</td>
<td>Bar3D</td>
</tr>
<tr>
<td>VCHART_3D_STACKED_BAR_ABSOLUTE</td>
<td>6</td>
<td>StackedBar3DAbsolute</td>
</tr>
<tr>
<td>VCHART_3D_STACKED_BAR_PERCENT</td>
<td>7</td>
<td>StackedBar3DPercent</td>
</tr>
<tr>
<td>VCHART_PIE</td>
<td>8</td>
<td>Pie</td>
</tr>
<tr>
<td>VCHART_3D_PIE</td>
<td>9</td>
<td>Pie3D</td>
</tr>
<tr>
<td>VCHART_AREA_ABSOLUTE</td>
<td>10</td>
<td>AreaAbsolute</td>
</tr>
<tr>
<td>VCHART_AREA_PERCENT</td>
<td>11</td>
<td>AreaPercent</td>
</tr>
</tbody>
</table>
Remarks:
VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see “Dynamic Positioning” in the Programmer's Manual.

Example ActiveX, .NET, Java, …:

```vbnet
Dim Data As TVPEChartData
Data = Doc.ChartDataCreate(2, 4)

creates a ChartData object with two columns (for example Apples and Bananas), where each column can hold 4 numeric data values (= rows).
```

```vbnet
data.AddValue(0, 10)
data.AddValue(0, 20)
data.AddValue(0, 30)
data.AddValue(0, 40)
data.AddValue(1, 5)
data.AddValue(1, 10)
data.AddValue(1, 15)
data.AddValue(1, 20)
data.AddLegend("Apples")
data.AddLegend("Bananas")
```

Now the internal data table of the Chart Data object will look like this:

<table>
<thead>
<tr>
<th></th>
<th>Column 0 &quot;Apples&quot;</th>
<th>Column 1 &quot;Bananas&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 0</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>row 1</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>row 2</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>row 3</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>
With the following code:

```plaintext
Doc.ChartXLabelState = ChartLabelState.User
Data.AddXLabel("1. Quarter")
Data.AddXLabel("2. Quarter")
Data.AddXLabel("3. Quarter")
Data.AddXLabel("4. Quarter")
```

the internal data table of the Chart Data object will look like this:

<table>
<thead>
<tr>
<th></th>
<th>Column 0</th>
<th>Column 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;Apples&quot;</td>
<td>&quot;Bananas&quot;</td>
</tr>
<tr>
<td>row 0</td>
<td>&quot;1. Quarter&quot;</td>
<td>10</td>
</tr>
<tr>
<td>row 1</td>
<td>&quot;2. Quarter&quot;</td>
<td>20</td>
</tr>
<tr>
<td>row 2</td>
<td>&quot;3. Quarter&quot;</td>
<td>30</td>
</tr>
<tr>
<td>row 3</td>
<td>&quot;4. Quarter&quot;</td>
<td>40</td>
</tr>
</tbody>
</table>

Now some different charts are created from the same ChartData object:

```plaintext
Doc.BkgMode = BkgMode.GradientLine
Doc.ChartTitle = "Bar Chart"
Doc.Chart(1, 1, -18, -18, Data, ChartType.Bar)
Doc.PageBreak()

Doc.ChartTitle = "3-D Bar Chart"
Doc.Chart(1, 1, -18, -18, Data, ChartType.Bar3D)
Doc.PageBreak()

Doc.ChartTitle = "3-D Pie Chart"
Doc.ChartSubTitle = "Row 1"
Doc.ChartRow = 0
Doc.Chart(1, 1, -18, -18, Data, ChartType.Pie3D)
Doc.PageBreak()

Doc.ChartTitle "3-D Pie Chart"
Doc.ChartSubTitle = "Row 2"
Doc.ChartRow = 1
Doc.Chart(1, 1, -18, -18, Data, ChartType.Pie3D)
```

Example VCL:

```plaintext
hData: Pointer
hData = Doc.ChartDataCreate(2, 4)
```

Creates a ChartData object with two columns (for example Apples and Bananas), where each column can hold 4 numeric data values (= rows).
Now the internal data table of the Chart Data object will look like this:

<table>
<thead>
<tr>
<th></th>
<th>Column 0 &quot;Apples&quot;</th>
<th>Column 1 &quot;Bananas&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 0</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>row 1</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>row 2</td>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>row 3</td>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>

With the following code

```vbe
Doc.ChartXLabelState = VCHART_LABEL_USER
Doc.ChartDataAddXLabel(hData, "1. Quarter")
Doc.ChartDataAddXLabel(hData, "2. Quarter")
Doc.ChartDataAddXLabel(hData, "3. Quarter")
Doc.ChartDataAddXLabel(hData, "4. Quarter")
```

the internal data table of the Chart Data object will look like this:

<table>
<thead>
<tr>
<th></th>
<th>Column 0 &quot;Apples&quot;</th>
<th>Column 1 &quot;Bananas&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 0</td>
<td>&quot;1. Quarter&quot;</td>
<td>10</td>
</tr>
<tr>
<td>row 1</td>
<td>&quot;2. Quarter&quot;</td>
<td>20</td>
</tr>
<tr>
<td>row 2</td>
<td>&quot;3. Quarter&quot;</td>
<td>30</td>
</tr>
<tr>
<td>row 3</td>
<td>&quot;4. Quarter&quot;</td>
<td>40</td>
</tr>
</tbody>
</table>
Now some different charts are created from the same ChartData object:

```plaintext
Doc.BkgMode = VBKG_GRD_LINE
Doc.ChartTitle = "Bar Chart"
Doc.Chart(1, 1, -18, -18, hData, VCHART_BAR)
Doc.PageBreak()

Doc.ChartTitle = "3-D Bar Chart"
Doc.Chart(1, 1, -18, -18, hData, VCHART_3D_BAR)
Doc.PageBreak()

Doc.ChartTitle = "3-D Pie Chart"
Doc.ChartSubTitle = "Row 1"
Doc.ChartRow = 0
Doc.Chart(1, 1, -18, -18, hData, VCHART_3D_PIE)
Doc.PageBreak()

Doc.ChartTitle = "3-D Pie Chart"
Doc.ChartSubTitle = "Row 2"
Doc.ChartRow = 1
Doc.Chart(1, 1, -18, -18, hData, VCHART_3D_PIE)
```
25 FormFields

25.1 FormFields

[Enterprise Edition and above]

You are familiar to such kind of fields from the many paper forms that are used today. For an overview and explanation, please see "FormFields" in the Programmer's Manual.
25.2 FormField

Creates a Form Field at the given position, using the number of character cells specified by CharCount. Accordingly, each character cell has the

width = ( (x2 - x) / CharacterCount) - max(DividerPenSize, AltDividerPenSize)

Example:
If x = 1 and x2 = 11, then the total width of the Form Field is 11cm - 1cm = 10cm. If CharacterCount = 10, then each character cell is 1 cm wide. Afterwards, VPE will subtract the pen size of the Divider or Alternative Divider - whichever is thicker - from the available cell width.

Additionally, the Form Field can compute a best matching font size itself (see FormFieldFlags - VEDIT_FLAG_AUTO_FONTSIZE): since VPE computes the width of a single character cell, it will search for a font size so that the widest character of the font will best fit into the available cell width and height.

If y2 = VFREE, the height of the FormField will be computed accordingly to the height of the currently selected font. Moreover, if y2 = VFREE and VEDIT_FLAG_AUTO_FONTSIZE is used, VPE will compute the height of the Form Field accordingly to the width of a single character cell with the same algorithm as explained above. When computing the height, VPE will also consider the thickness of the bottom line, of course.

method VpeCoord VPE.FormField(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    string Text
)

VpeCoord Left, Top, Right, Bottom
    position and dimensions of the FormField

string Text
    the content string of the FormField

Returns:
    the bottom y-coordinate generated by the output

Remarks:
- Do not misunderstand FormFields, they can't be used for data input by the user. They are intended to be used to display data. (The VPE Interactive Edition offers objects for data input.)
- A Form Field can only consist of one single line of text, it can not have multiple lines.
- The text alignment can only be left or right.
- Form Fields can not be rotated.
- Form Fields do not fire AutoBreaks.
- If x2 = VFREE, the FormField will create a normal Plain Text Object as if Write(Box) was used.
• The bottom line will only be drawn, if the normal PenSize (used for lines, frames, etc.) is zero. If PenSize > 0, a FormField will have instead a surrounding frame (with the thickness of PenSize) as with WriteBox().

• VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

See also:
  RenderFormField
  "FormFields" in the Programmer's Manual
25.3 CharCount

Sets the maximum number of characters that can be displayed in a FormField. This value equals the number of character cells drawn in a FormField.

[Interactive Edition only]
if the value of CharCount is negative, an Interactive FormField is painted without dividers as if CharCount was zero, but the absolute value of CharCount specifies the maximum number of characters, which may be entered into the control by the user.
e.g. CharCount = -15 would mean, that a user may only enter up to 15 characters into the Interactive FormField.

property long VPE.CharCount
read / write; runtime only; also supported by TVPEObject

Possible Values:
The number of character cells drawn in a FormField

Default:
15

Remarks:
If CharCount is set to zero, the FormField will behave the same as if Write(Box) was used.

See also:
"FormFields" in the Programmer's Manual
25.4 DividerPenSize

Sets the pen size for the dividers of a FormField.

**property VpeCoord VPE.DividerPenSize**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**
the pen size

**Default:**
0.01 ( = 0.1mm)

**Remarks:**
The pen size influences the available width of each character cell. If the font size is computed automatically (see FormFieldFlags), the value for the pen size influences the computed font size.

**See also:**
"FormFields" in the Programmer's Manual
25.5 DividerPenColor
Sets the pen color for the dividers of a FormField.

property Color VPE.DividerPenColor
    read / write; runtime only; also supported by TVPEObject

Possible Values:
    any of the “COLOR_xyz” constants described in Programmer’s Manual
    or ActiveX / VCL: any RGB value
    or .NET: any member of the .NET Color structure

Default:
    COLOR_BLACK

See also:
    "FormFields" in the Programmer's Manual
25.6 AltDividerNPosition
Sets the position where Alternative Dividers are drawn instead of normal Dividers.

property long VPE.AltDividerNPosition
    read / write; runtime only; also supported by TVPEObject

Possible Values:
the position of Alternative Dividers

Default:
0 (no Alternative Dividers are drawn)

See also:
"FormFields" in the Programmer's Manual
25.7 AltDividerPenSize

Sets the pen size for the Alternative Dividers of a FormField.

**property VpeCoord VPE.AltDividerPenSize**

   read / write; runtime only; also supported by TVPEObject

**Possible Values:**
the pen size

**Default:**
0.03 ( = 0.3mm)

**Remarks:**
The pen size influences the available width of each character cell. If the font size is computed automatically (see FormFieldFlags), the value for the pen size influences the computed font size.

**See also:**
"FormFields" in the Programmer's Manual
25.8 AltDividerPenColor

Sets the pen color for the Alternative Dividers of a FormField.

**property Color VPE.AltDividerPenColor**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**
any of the "COLOR_xyz" constants described in Programmer's Manual
or **ActiveX / VCL:** any RGB value
or **.NET:** any member of the .NET Color structure

**Default:**
COLOR_BLACK

**See also:**
"FormFields" in the Programmer's Manual
25.9 BottomLinePenSize

Sets the pen size for the bottom line. A bottom line is only drawn, if the global PenSize (used for lines, frames, etc.) is set to zero. Otherwise a framed FormField will be drawn similar to WriteBox().

property VpeCoord VPE.BottomLinePenSize

    read / write; runtime only; also supported by TVPEObject

Possible Values:
    the pen size for the bottom line

Default:
    0.03 ( = 0.3 mm)

See also:
    "FormFields" in the Programmer's Manual
25.10 BottomLinePenColor

Sets the pen color for the bottom line. A bottom line is only drawn, if the global PenSize is set to zero. Otherwise a framed FormField will be drawn similar to WriteBox().

property Color VPE.BottomLinePenColor

    read / write; runtime only; also supported by TVPEObject

Possible Values:
    any of the "COLOR_xyz" constants described in Programmer's Manual
    or ActiveX / VCL: any RGB value
    or .NET: any member of the .NET Color structure

Default:
    COLOR_BLACK

See also:
    "FormFields" in the Programmer's Manual
25.11 DividerStyle

Sets the paint-style for dividers.

**property DividerStyle [long] VPE.DividerStyle**

read / write; runtime only; also supported by TVPEObject

**Possible Values:**
the style of dividers, possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFF_STYLE_NONE</td>
<td>0</td>
<td>StyleNone</td>
<td>no dividers</td>
</tr>
<tr>
<td>VFF_STYLE_1_4</td>
<td>1</td>
<td>Style_1_4</td>
<td>1/4 height</td>
</tr>
<tr>
<td>VFF_STYLE_1_3</td>
<td>2</td>
<td>Style_1_3</td>
<td>1/3 height</td>
</tr>
<tr>
<td>VFF_STYLE_1_2</td>
<td>3</td>
<td>Style_1_2</td>
<td>1/2 height (default)</td>
</tr>
<tr>
<td>VFF_STYLE_2_3</td>
<td>4</td>
<td>Style_2_3</td>
<td>2/3 height</td>
</tr>
<tr>
<td>VFF_STYLE_3_4</td>
<td>5</td>
<td>Style_3_4</td>
<td>3/4 height</td>
</tr>
<tr>
<td>VFF_STYLE_1_1</td>
<td>6</td>
<td>Style_1_1</td>
<td>full height (default for AltDivider)</td>
</tr>
</tbody>
</table>

**Default:**
VFF_STYLE_1_2, dividers are drawn at ½ height of the FormField's height

**See also:**
"FormFields" in the Programmer's Manual
25.12 AltDividerStyle
Sets the paint-style for Alternative Dividers.

property DividerStyle [long] VPE.AltDividerStyle
    read / write; runtime only; also supported by TVPEObject

Possible Values:
the style of Alternative Dividers, possible values are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFF_STYLE_NONE</td>
<td>0</td>
<td>StyleNone</td>
<td>no dividers</td>
</tr>
<tr>
<td>VFF_STYLE_1_4</td>
<td>1</td>
<td>Style_1_4</td>
<td>1/4 height</td>
</tr>
<tr>
<td>VFF_STYLE_1_3</td>
<td>2</td>
<td>Style_1_3</td>
<td>1/3 height</td>
</tr>
<tr>
<td>VFF_STYLE_1_2</td>
<td>3</td>
<td>Style_1_2</td>
<td>1/2 height (default)</td>
</tr>
<tr>
<td>VFF_STYLE_2_3</td>
<td>4</td>
<td>Style_2_3</td>
<td>2/3 height</td>
</tr>
<tr>
<td>VFF_STYLE_3_4</td>
<td>5</td>
<td>Style_3_4</td>
<td>3/4 height</td>
</tr>
<tr>
<td>VFF_STYLE_1_1</td>
<td>6</td>
<td>Style_1_1</td>
<td>full height (default for AltDivider)</td>
</tr>
</tbody>
</table>

Default:
VFF_STYLE_1_1, Alternative Dividers are drawn at the full height of the FormField's height

See also:
"FormFields" in the Programmer's Manual
25.13 FormFieldFlags
Sets additional control flags for FormFields.

**property FormFieldFlags [long] VPE.FormFieldFlags**
read / write; runtime only; also supported by TVPEObject

**Possible Values:**
possible values are any combination of:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFF_FLAG_DIV_FIRST</td>
<td>1</td>
<td>DivFirst</td>
<td>first divider is painted (only, if no frame)</td>
</tr>
<tr>
<td>VFF_FLAG_DIV_LAST</td>
<td>2</td>
<td>DivLast</td>
<td>last divider is painted (only, if no frame)</td>
</tr>
<tr>
<td>VFF_FLAG_ALT_FIRST</td>
<td>4</td>
<td>AltFirst</td>
<td>(default) first divider is painted as AltDivider (overrides _DIV_FIRST) (only, if no frame)</td>
</tr>
<tr>
<td>VFF_FLAG_ALT_LAST</td>
<td>8</td>
<td>AltLast</td>
<td>(default) last divider is painted as AltDivider (overrides _DIV_LAST) (only, if no frame)</td>
</tr>
<tr>
<td>VFF_FLAG_AUTO_FONTSIZE</td>
<td>16</td>
<td>AutoFontSize</td>
<td>(default) font size is computed automatically</td>
</tr>
</tbody>
</table>

**Default:**
VFF_FLAG_ALT_FIRST + VFF_FLAG_ALT_LAST + VFF_FLAG_AUTO_FONTSIZE

**Remarks:**
You can use combinations of the above flags by adding their values. Please note that if you use VFF_FLAG_ALT_FIRST, VFF_FLAG_DIV_FIRST is ignored. Also if you use VFF_FLAG_ALT_LAST, VFF_FLAG_DIV_LAST is ignored.

If you specify VFF_FLAG_AUTO_FONTSIZE, the font size for the FormField is computed automatically according to the available space for a single character. See FormField for details.

**See also:**
"FormFields" in the Programmer's Manual
26 Template Functions

26.1 Template Functions
[Enterprise Edition and above]
This section describes the methods and properties related to Templates.

See also:
- *dycodoc* Template Processing in the Programmer's Manual
26.2 LoadTemplate

Loads a template file and creates a Template Object from it in memory.

The method returns a Template Object, which is the anchor object for the other template processing functions.

Using the VPE API, you can query the layout- and data source structure from a template. In addition you are able to perform several operations and modifications on a Template Object by code, including the assignment of values to fields (i.e. variables).

The template is logically linked to the specified VPE document and it will be removed automatically from memory when the VPE document is closed. Of course the template will be removed latest from memory when your application terminates.

```csharp
method TVPETemplate VPE.LoadTemplate(
    string FileName
)
```

`string FileName`

the path and file name of the file to load

**Returns:**

The Template Object, if the template file was found and could be loaded.

In case of an error an exception is thrown and `LastError` is set.

**Remarks:**

In case of an error, LastError is set accordingly to the status of the load process.

If the template has assigned an Authenticity Key, this method will fail. Use `LoadTemplateAuthKey()` for loading templates with an Authenticity Key.

**See also:**

dycodoc Template Processing in the Programmer’s Manual
26.3 LoadTemplateAuthKey

This method is identical to `LoadTemplate()`, except that it validates the *Authenticity Key* which is assigned to a template by using *dycodoc*.

Loads a template file and creates a Template Object from it in memory.

The method returns a Template Object, which is the anchor object for the other template processing functions.

Using the VPE API, you can query the layout- and data source structure from a template. In addition you are able to perform several operations and modifications on a Template Object by code, including the assignment of values to fields (i.e. variables).

The template is logically linked to the specified VPE document and it will be removed automatically from memory when the VPE document is closed. Of course the template will be removed latest from memory when your application terminates.

```csharp
method TVPETemplate VPE.LoadTemplateAuthKey(
    string FileName,
    long Key1,
    long Key2,
    long Key3,
    long Key4
)
```

*string FileName*
the path and file name of the file to load

*long Key1, Key2, Key3, Key4*
the Authenticity Key

**Returns:**
The Template Object, if the template file was found, could be loaded and the Authenticity Key validation was successful, i.e. the Authenticity Key is present and checks ok.
In case of an error an exception is thrown and `LastError` is set.

**Remarks:**
In case of an error, LastError is set accordingly to the status of the load process.

**See also:**
*dycodoc* Template Processing in the Programmer's Manual
Validating the Template Authenticity Key in the Programmer's Manual
26.4 DumpTemplate
After the values for all required Fields have been set, the whole template is dumped into the current and all following pages of the VPE document with this method.

```c
method integer VPE.DumpTemplate(
    TVPETemplate objTemplate
)
```

TVPETemplate objTemplate
the Template Object that shall be dumped

Returns:
Error Status, this is either VERR_OK if no error occurred, or any of the VERR_xyz values.

Remarks:
The DumpTemplate()-methods do not make any differences between the AutoBreakMode AUTO_BREAK_ON and AUTO_BREAK_FULL, i.e. objects are always inserted into the current VPE Document with the given left and right coordinates if one of either modes is set.

In case of an error, LastError is set.

A template may only be dumped into the VPE document into which it was loaded, otherwise LastError will be set to VERR_TPL.OwnerShip.

[Interactive Edition only]
A template which contains Controls can only be dumped once into a VPE Document. If you wish to dump it more than once into one and the same VPE Document, you need to load the same template as often into memory as you want to dump it. Otherwise DumpTemplate() will return the error code VERR_TPL_PAGE_ALREADY_DUMPED. A template which does not contain Controls can be dumped as often into one and the same VPE Document as you like.

See also:
dycodoc Template Processing in the Programmer's Manual
26.5 DumpTemplatePage

After the values for all required Fields have been set, the template page specified by the parameter "page" is dumped into the current and - if an auto break occurs - all following pages of the VPE document with this method.

```plaintext
method integer VPE.DumpTemplatePage(
    TVPETemplate objTemplate,
    long page
)
```

- **TVPETemplate objTemplate**
  - the Template Object that shall be dumped

- **long page**
  - The page of the template that shall be dumped, may be in the range of 0 (= first page) to PageCount - 1 (= last page).

**Returns:**
- Error Status, this is either VERR_OK if no error occurred, or any of the VERR_xyz values.

**Remarks:**
- The `DumpTemplate()`-methods do not make any differences between the `AutoBreakMode AUTO_BREAK_ON` and `AUTO_BREAK_FULL`, i.e. objects are always inserted into the current VPE Document with the given left and right coordinates if one of either modes is set.

  In case of an error, `LastError` is set.

  A template may only be dumped into the VPE document into which it was loaded, otherwise `LastError` will be set to VERR_TPL_OWNERSHIP.

**[Interactive Edition only]**

  A template which contains Controls can only be dumped once into a VPE Document. If you wish to dump it more than once into one and the same VPE Document, you need to load the same template as often into memory as you want to dump it. Otherwise `DumpTemplate()` will return the error code VERR_TPL_PAGE_ALREADY_DUMPED. A template which does not contain Controls can be dumped as often into one and the same VPE Document as you like.

**See also:**
- `dycodoc` Template Processing in the Programmer's Manual
26.6 UseTemplateMargins

Retrieves the margin settings from the specified page out of the template and sets the margins of the VPE document correspondingly for the current page only. This function is useful if you need to copy margin settings from a template to a page of a VPE document, e.g. because you create some pages in addition to the template by code using the VPE API.

```
method void VPE.UseTemplateMargins(
    TVPETemplate objTemplate,
    long page
)
```

```
long page
```

The page of the template whose margins shall be retrieved, may be in the range of 0 (= first page) to PageCount - 1 (= last page).

See also:
- dycodoc Template Processing in the Programmer's Manual
26.7 UseTemplateSettings

Retrieves the whole page settings (page dimensions, margins, orientation, paper bin) from the specified page of the template and sets them correspondingly in the VPE document for the current page only.

This function is useful if you need to copy page settings from a template to a page of a VPE document, e.g. because you create some pages in addition to the template by code using the VPE API.

```csharp
method void VPE.UseTemplateSettings(
    TVPETemplate objTemplate,
    long page
)
```

`long page`

The page of the template whose page settings shall be retrieved, may be in the range of 0 (= first page) to PageCount - 1 (= last page).

See also:

dycodoc Template Processing in the Programmer’s Manual
27 Template Object

27.1 Template Object - TVPETemplate

[Enterprise Edition and above]
This section describes the methods and properties of the Template Object.
The class-name is TVPETemplate.

.NET: The TVPETemplate class is derived from the base class Object.

See also:
dycodoc Template Processing in the Programmer's Manual
27.2 ObjectHandle

This is the internal object handle of the VPE DLL for the Template Object.
Do not use.

property long TVPETemplate.ObjectHandle

    read; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>not initialized</td>
</tr>
<tr>
<td>NON-NULL</td>
<td>it holds a reference to an internal Template Object in the VPE DLL</td>
</tr>
</tbody>
</table>

See also:

dycodoc Template Processing in the Programmer's Manual
27.3 Master

If the page settings (page dimensions, margins, orientation, paper bin) of the template shall be used when dumping the whole template or a single page of the template into a VPE document, specify True for the parameter yes_no, False otherwise.

If you are using a template which is not defined as master, strange results can happen while dumping, if the margins or page dimensions of the VPE document differ from the template, because it can cause unwanted page breaks (objects in the template are outside of the VPE document margins, and therefore an auto page break occurs.)

**property boolean [integer] TVPETemplate.Master**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, the template's page settings are used</td>
</tr>
<tr>
<td>False</td>
<td>no, the template's page settings are not used</td>
</tr>
</tbody>
</table>

**Default:**

True = yes, this template is master

(this is the default after a template has been loaded into memory using `LoadTemplate()`)

**Remarks:**

In dycodoc, objects cannot be made dependent if they exist on different pages. Only objects on one and the same page can be made dependent. It is very useful to set Master = False if you wish to have a lot of objects to be dependent, which do not fit onto a single page in dycodoc. Solution: Create a very long page in dycodoc and place the objects on it. Load this page into VPE (which has a smaller page format defined) and set the `Master` property of the template to False: the objects that do not fit onto the current page are automatically broken to the next page(s).

**See also:**

* dycodoc Template Processing in the Programmer's Manual
27.4 ClearFields
Clears all Fields of a template, i.e. null-values are assigned to all fields within the given template.

method void TVPETemplate.ClearFields()

Remarks:
A null-value is a special value, like in SQL databases. An empty string or an integer / double / date with the value zero are not null-values.

In string expressions a null-value is treated like an empty string. In numeric or date expressions, a null-value is treated like a zero.

See also:
dycodoc Template Processing in the Programmer's Manual
27.5 GetFieldIsNull

Tests whether a Field has a null-value. A field has a null-value after calling ClearFields or after explicitly assigning a null-value to a Field by either calling TVPEField.SetNull or TVPETemplate.SetFieldToNull.

```
method boolean TVPETemplate.GetFieldIsNull(
    string DataSourcePrefix,
    string FieldName
)
```

- **string DataSourcePrefix**: prefix of the Data Source, this is the unique name of a Data Source as defined in dydoc

- **string FieldName**: name of the Data Source Field

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>the field has a null-value</td>
</tr>
<tr>
<td>False</td>
<td>the field has not a null-value or the field was not found</td>
</tr>
</tbody>
</table>

See also:

dydoc Template Processing in the Programmer's Manual
27.6 SetFieldToNull
Assigns a null-value to a field. A field has a null-value after calling ClearFields or after explicitly assigning a null-value to a Field by either calling TVPEField.SetNull or by calling this method.

**method** void TVPETemplate.SetFieldToNull(
    string DataSourcePrefix,
    string FieldName
)

*string DataSourcePrefix*
prefix of the Data Source, this is the unique name of a Data Source as defined in *dycodoc*

*string FieldName*
name of the Data Source Field

**See also:**
*dycodoc* Template Processing in the Programmer's Manual
27.7 GetFieldNullValueText

Returns the text which is printed for Fields that have a null-value. You can assign a special text to Fields with a null-value. By default, the null-value text is an empty string. You can specify for each field an individual null-value text, for example "n/a".

```csharp
method boolean TVPETemplate.GetFieldNullValueText(
    string DataSourcePrefix,
    string FieldName
)
```

- **string DataSourcePrefix**
  - prefix of the Data Source, this is the unique name of a Data Source as defined in *dycodec*

- **string FieldName**
  - name of the Data Source Field

**Returns:**
- The string with the null-value text of the Field.

**Remarks:**
- In case of an error, an exception is thrown (for example, the field was not found, or out of memory)

**See also:**
27.8 SetFieldNullValueText

Sets the text which is printed for Fields that have a null-value. You can assign a special text to Fields with a null-value. By default, the null-value text is an empty string. You can specify for each field an individual null-value text, for example "n/a".

```csharp
method boolean TVPETemplate.SetFieldNullValueText(
    string DataSourcePrefix,
    string FieldName,
    string Text
)
```

- **string DataSourcePrefix**
  - prefix of the Data Source, this is the unique name of a Data Source as defined in `dycodoc`
- **string FieldName**
  - name of the Data Source Field
- **string Text**
  - the string that shall be assigned to the Field's null-value text

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>if the field was found and its null-value text could be set</td>
</tr>
<tr>
<td>False</td>
<td>otherwise</td>
</tr>
</tbody>
</table>

**See also:**

`dycodoc` Template Processing in the Programmer's Manual
27.9 GetFieldAsString

Returns the value of a Field as string.

```csharp
method string TVPETemplate.GetFieldAsString(
    string DataSourcePrefix,
    string FieldName
)
```

- **string DataSourcePrefix**: prefix of the Data Source, this is the unique name of a Data Source as defined in *dycodec*
- **string FieldName**: name of the Data Source Field

**Returns:**

The string value of the Field.

**Remarks:**

In case of an error, an exception is thrown (for example, the field was not found, or out of memory)

**See also:**

*dycodec* Template Processing in the Programmer’s Manual
27.10 SetFieldAsString

Sets the value of a Field as string.

```csharp
method int TVPETemplate.SetFieldAsString(
    string DataSourcePrefix,
    string FieldName,
    string Value
)
```

*string DataSourcePrefix*
prefix of the Data Source, this is the unique name of a Data Source as defined in *dycodoc*

*string FieldName*
name of the Data Source Field

*string Value*
the string that shall be assigned to the Field

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>if the field was found and its value could be set</td>
</tr>
<tr>
<td>False</td>
<td>otherwise</td>
</tr>
</tbody>
</table>

**See also:**

dycodoc Template Processing in the Programmer's Manual
27.11 GetFieldAsInteger

Returns the value of a Field as integer.

```csharp
method integer TVPETemplate.GetFieldAsInteger(
    string DataSourcePrefix,
    string FieldName
)
```

- **string DataSourcePrefix**
  - prefix of the Data Source, this is the unique name of a Data Source as defined in *dycodoc*

- **string FieldName**
  - name of the Data Source Field

**Returns:**
- The integer value of the Field.

**Remarks:**
- In case of an error, an exception is thrown (for example, if the field contained pure text that could not be converted to an integer value)

**See also:**
- *dycodoc* Template Processing in the Programmer's Manual
27.12 SetFieldAsInteger

Sets the value of a Field as integer.

**method integer TVPETemplate.SetFieldAsInteger(**
  string **DataSourcePrefix**,
  string **FieldName**,
  integer **Value**
**)

**string DataSourcePrefix**
  prefix of the Data Source, this is the unique name of a Data Source as defined in *dycodoc*

**string FieldName**
  name of the Data Source Field

**integer Value**
  the integer value that shall be assigned to the Field

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>if the field was found and its value could be set</td>
</tr>
<tr>
<td>False</td>
<td>otherwise</td>
</tr>
</tbody>
</table>

**See also:**

*dycodoc* Template Processing in the Programmer's Manual
27.13 GetFieldAsNumber

Returns the value of a Field as number (double).

```csharp
method double TVPETemplate.GetFieldAsNumber(
    string DataSourcePrefix,
    string FieldName
)
```

- **string DataSourcePrefix**
  - prefix of the Data Source, this is the unique name of a Data Source as defined in dycodoc

- **string FieldName**
  - name of the Data Source Field

**Returns:**
- The double value of the Field.

**Remarks:**
- In case of an error, an exception is thrown (for example, if the field contained pure text that could not be converted to a number value)

**See also:**
- dycodoc Template Processing in the Programmer's Manual
27.14 SetFieldAsNumber
Sets the value of a Field as number (double).

```csharp
method integer TVPETemplate.SetFieldAsNumber(
    string DataSourcePrefix,
    string FieldName,
    double Value
)
```

- `string DataSourcePrefix` - prefix of the Data Source, this is the unique name of a Data Source as defined in *dycodoc*
- `string FieldName` - name of the Data Source Field
- `double Value` - the double value that shall be assigned to the Field

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>if the field was found and its value could be set</td>
</tr>
<tr>
<td>False</td>
<td>otherwise</td>
</tr>
</tbody>
</table>

**See also:**
dycodoc Template Processing in the Programmer's Manual
27.15 GetFieldAsBoolean

Returns the value of a Field as boolean.

```java
method boolean [integer] TVPETemplate.GetFieldAsBoolean(
    string DataSourcePrefix,
    string FieldName
)
```

- **string DataSourcePrefix**: prefix of the Data Source, this is the unique name of a Data Source as defined in `dycodoc`
- **string FieldName**: name of the Data Source Field

**Returns:**
- The boolean value of the Field.

**Remarks:**
In case of an error, an exception is thrown (for example, if the field contained pure text that could not be converted to a boolean value)

**See also:**
- `dycodoc` Template Processing in the Programmer's Manual
### 27.16 SetFieldAsBoolean

Sets the value of a Field as boolean.

```csharp
method integer TVPETemplate.SetFieldAsNumber(
    string DataSourcePrefix,
    string FieldName,
    boolean Value
)
```

- **string DataSourcePrefix**
  - prefix of the Data Source, this is the unique name of a Data Source as defined in `dycodoc`

- **string FieldName**
  - name of the Data Source Field

- **boolean Value**
  - the boolean value that shall be assigned to the Field

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>if the field was found and its value could be set</td>
</tr>
<tr>
<td>False</td>
<td>otherwise</td>
</tr>
</tbody>
</table>

**See also:**

`dycodoc` Template Processing in the Programmer's Manual
27.17 DateTimeIsUTC

Specifies, whether VPE’s date/time functions receive and return date/time values in UTC (Universal Time Coordinated) or in local time.

**property boolean [integer] VPE. DateTimeIsUTC**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, parameters and return values of VPE’s date/time functions are in UTC</td>
</tr>
<tr>
<td>False</td>
<td>no, parameters and return values of VPE’s date/time functions are in local time</td>
</tr>
</tbody>
</table>

**Default:**

False = parameters and return values of VPE’s date/time functions are in local time

**Example:**

```
Doc.DateTimeIsUTC = true
```

**See also:**

dycodoc Template Processing in the Programmer’s Manual
27.18 SetFieldAsDateTime

Sets the value of a Field as date / time.

```plaintext
method integer TVPETemplate.SetFieldAsDateTime(
    string DataSourcePrefix,
    string FieldName,
    integer year,
    integer month,
    integer day,
    integer hour,
    integer minute,
    integer second,
    integer msec
)

string DataSourcePrefix
    prefix of the Data Source, this is the unique name of a Data Source as defined in dycodoc

string FieldName
    name of the Data Source Field

int year, month, day, hour, minute, second, msec
    the integer values, which shall be assigned to the respective values of the Field as integer

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>if the field was found and its value could be set</td>
</tr>
<tr>
<td>False</td>
<td>otherwise</td>
</tr>
</tbody>
</table>

See also:

dycodoc Template Processing in the Programmer's Manual
27.19 GetFieldAsOleDateTime

[not in .NET]

Returns the value of a Field as OLE date / time.

```csharp
method date TVPETemplate.GetFieldAsOleDateTime(
    string DataSourcePrefix,
    string FieldName
)
```

string DataSourcePrefix  
prefix of the Data Source, this is the unique name of a Data Source as defined in dycodoc

string FieldName  
name of the Data Source Field

**Returns:**
The OLE date / time value of the Field.

**Remarks:**
In case of an error, an exception is thrown (for example, if the field contained pure text that could not be converted to an OLE date / time value)

**See also:**
dycodoc Template Processing in the Programmer's Manual
27.20 SetFieldAsOleDateTime

[not in .NET]
Sets the value of a Field as OLE date / time.

method integer TVPETemplate.SetFieldAsOleDateTime(
    string DataSourcePrefix,
    string FieldName,
    date Value
)

string DataSourcePrefix
    prefix of the Data Source, this is the unique name of a Data Source as defined in dycodoc

string FieldName
    name of the Data Source Field

date Value
    the OLE date / time value that shall be assigned to the Field

Returns:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>if the field was found and its value could be set</td>
</tr>
<tr>
<td>False</td>
<td>Otherwise</td>
</tr>
</tbody>
</table>

See also:

dycodoc Template Processing in the Programmer's Manual
27.21 GetFieldAsNetDateTime

Returns the value of a Field as .NET date / time.

**method DateTime TVPETemplate.GetFieldAsNetDateTime(**

```
    string DataSourcePrefix,
    string FieldName
```

**)

**string DataSourcePrefix**

prefix of the Data Source, this is the unique name of a Data Source as defined in dydoc

**string FieldName**

name of the Data Source Field

**Returns:**

The .NET date / time value of the Field.

**Remarks:**

In case of an error, an exception is thrown (for example, if the field contained pure text that could not be converted to a .NET date / time value)

**See also:**

dydoc Template Processing in the Programmer's Manual
27.22 SetFieldAsNetDateTime

Sets the value of a Field as .NET date / time.

```csharp
method integer TVPETemplate.SetFieldAsNetDateTime(
    string DataSourcePrefix,
    string FieldName,
    DateTime Value
)
```

- **string DataSourcePrefix**
  - prefix of the Data Source, this is the unique name of a Data Source as defined in *dycodoc*

- **string FieldName**
  - name of the Data Source Field

- **DateTime Value**
  - the .NET date / time value that shall be assigned to the Field

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>If the field was found and its value could be set</td>
</tr>
<tr>
<td>False</td>
<td>otherwise</td>
</tr>
</tbody>
</table>

**See also:**

*dycodoc* Template Processing in the Programmer's Manual
27.23 GetFieldAsJavaDateTime

Returns the value of a Field as Java date / time (number of milliseconds since midnight Jan 1, 1970).

```java
method double TVPETemplate.GetFieldAsJavaDateTime(
    string DataSourcePrefix,
    string FieldName
)
```

**string DataSourcePrefix**
- prefix of the Data Source, this is the unique name of a Data Source as defined in dycodoc

**string FieldName**
- name of the Data Source Field

**Returns:**
- The Java date / time value of the Field.

**Remarks:**
- In case of an error, an exception is thrown (for example, if the field contained pure text that could not be converted to an Java date / time value)

**See also:**
- dycodoc Template Processing in the Programmer's Manual
27.24 SetFieldAsJavaDateTime

Sets the value of a Field as Java date / time (number of milliseconds since midnight Jan 1, 1970).

```java
method integer TVPETemplate.SetFieldAsJavaDateTime(
    string DataSourcePrefix,
    string FieldName,
    double Value
)
```

string **DataSourcePrefix**

prefix of the Data Source, this is the unique name of a Data Source as defined in *dycodec*

string **FieldName**

name of the Data Source Field

double **Value**

the Java date / time value that shall be assigned to the Field

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>if the field was found and its value could be set</td>
</tr>
<tr>
<td>False</td>
<td>otherwise</td>
</tr>
</tbody>
</table>

**See also:**

dycodec Template Processing in the Programmer's Manual
27.25 ClearFields

Resets all Fields of the specified template to NULL.

method void TVPETemplate.ClearFields( )

See also:
  dycodoc Template Processing in the Programmer's Manual
27.26 DataSourceCount

Returns the total number of Data Sources used within the given template.

**property long TVPETemplate.DataSourceCount**

read; runtime only

**Returns:**

the total number of Data Sources used within the given template

**See also:**

dycodoc Template Processing in the Programmer's Manual
27.27 DataSourceObject

Returns a Data Source Object.

**method TVPEDataSource TVPETemplate.DataSourceObject(**

  long DataSourceIndex

  )

*long DataSourceIndex*

  index into the array of available Data Sources

  this value must be in the range between 0 and DataSourceCount - 1

**Returns:**

  the Data Source Object; if DataSourceIndex is out of range, an exception is thrown

**Remarks:**

  The function can be used to access Data Source Objects directly.

**See also:**

dycodoc Template Processing in the Programmer's Manual
Important Note for VPE-VCL Users in the Programmer's Manual
27.28 FindFieldObject

Searches for a given Field and returns the Field object.

```csharp
method TVPEField TVPETemplate.FindFieldObject(
    string DataSourcePrefix,
    string FieldName
)

string DataSourcePrefix
    prefix of the Data Source, this is the unique name of a Data Source as defined in dycodoc

string FieldName
    name of the Data Source Field

Returns:
    a Field Object; if the field is not found the ObjectHandle of the returned object is null.

Remarks:
    The function can be used to access Field Objects directly.

Example:

    template.FindFieldObject(hTpl, "SomeTable", "Year").AsInteger = 1982;

See also:
    dycodoc Template Processing in the Programmer's Manual
    Important Note for VPE-VCL Users in the Programmer's Manual
27.29 PageCount

Returns the number of pages in the given template.

property long TVPETemplate.PageCount

   read; runtime only

Returns:

   the number of pages in the given template

See also:

   dycodoc Template Processing in the Programmer's Manual
27.30 PageObject

Returns the Page Object of the specified page.

**method TVPETemplatePage TVPETemplate.PageObject(**

```c++
    long PagelIndex
```

**Returns:**

the Page Object; if PagelIndex is out of range an exception is thrown

**Remarks:**

The function can be used to access Page Objects directly.

**See also:**

dycodec Template Processing in the Programmer's Manual
Important Note for VPE-VCL Users in the Programmer's Manual
27.31 FindVpeObject

In *dycodoc* a unique name is assigned to each **VPE Object**. This function searches for a given VPE Object name in the template and returns the VPE Object.

**method TVPEObject TVPETemplate.FindVpeObject(**
  string ObjectName
**)

**string ObjectName**

name of the VPE Object as defined in *dycodoc*

**Returns:**

a VPE Object. If the object is not found the ObjectHandle of the returned object is null.

**Remarks:**

The function can be used to access VPE Objects within the template directly.

**Example:**

**ActiveX / VCL:**

```vbnet
Dim VPEObject as TVPEObject
VPEObject = template.FindVpeObject("Text1")
VPEObject.BkgMode = VBKG_SOLID
VPEObject.BkgColor = COLOR_RED
```

**.NET:**

```csharp
Dim VPEObject as TVPEObject
VPEObject = template.FindVpeObject("Text1")
VPEObject.BkgMode = BkgMode.Solid
VPEObject.BkgColor = Color.Red
```

The above example searches for the VPE Object named "Text1" in the template and changes its background mode to solid and its color to red.

**See also:**

*dycodoc* Template Processing in the Programmer’s Manual

Important Note for VPE-VCL Users in the Programmer's Manual
28 DataSource Object

28.1 DataSource Object - TVPEDataSource

[Enterprise Edition and above]
This section describes the methods and properties of the Data Source Object. A Data Source encapsulates a table, which is defined in the dycodoc field definitions pane.

The class-name is TVPEDataSource.

.NET: The TVPEDataSource class is derived from the base class Object.

See also:

dycodoc Template Processing in the Programmer's Manual
28.2 ObjectHandle

This is the internal object handle of the VPE DLL for the DataSource Object.

The internal Object Handle is of interest to check if its value is null, which means that it is not initialized, i.e. it has no corresponding object in the DLL. This can happen in case of errors, i.e. if the object is not existing.

property long TVPEDataSource.ObjectHandle

read; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>not initialized</td>
</tr>
<tr>
<td>NON-NULL</td>
<td>it holds a reference to an internal object in the VPE DLL</td>
</tr>
</tbody>
</table>

See also:

dycodoc Template Processing in the Programmer's Manual
28.3 Prefix
Returns the prefix of the Data Source, i.e. the table name as defined in \textit{dycodoc}.

\textbf{property string TVPEDataSource.Prefix}
read; runtime only

\textbf{Returns:}
the prefix of the Data Source as defined in \textit{dycodoc}

\textbf{See also:}
d\textit{codo}c Template Processing in the Programmer's Manual
28.4 FileName

Returns the file name of the Data Source. This is the same name as the table name.

property string TVPDataSource.FileName

read; runtime only

**Returns:**
the file name of the Data Source

**See also:**
dycodoc Template Processing in the Programmer's Manual
28.5 Description
Returns the description of the Data Source as defined in *dycodoc*.

**property string TVPEDataSource.Description**
read; runtime only

**Returns:**
the description of the Data Source as defined in *dycodoc*

**See also:**
*dycodoc* Template Processing in the Programmer's Manual
28.6 FieldCount

Returns the total number of Fields used within the Data Source.

**property long TVPDataSource.FieldCount**

read; runtime only

**Returns:**

the total number of Fields used within the Data Source

**See also:**

dycodoc Template Processing in the Programmer's Manual
28.7 FieldObject

Returns a Field Object.

```c
method TVPField TVPDataSource.FieldObject(
    long FieldIndex
)
```

`long FieldIndex`
index into the array of available Fields
this value must be in the range between 0 and `FieldCount()` - 1

**Returns:**
a Field Object

**Remarks:**
The function can be used to access Field Objects within the Data Source directly.

**See also:**
dycodoc Template Processing in the Programmer's Manual
Important Note for VPE-VCL Users in the Programmer's Manual
29 Field Object

29.1 Field Object - TVPEField

[Enterprise Edition and above]
This section describes the methods and properties of the Field Object. A Field Object encapsulates a single field of a Data Source / table.

The class-name is TVPEField.

.NET: The TVPEField class is derived from the base class Object.

See also:

dycodoc Template Processing in the Programmer's Manual
29.2 ObjectHandle

This is the internal object handle of the VPE DLL for the Field Object.

The internal Object Handle is of interest to check if its value is null, which means that it is not initialized, i.e. it has no corresponding object in the DLL. This can happen in case of errors, i.e. if the object is not existing.

**property long TVPEField.ObjectHandle**

read; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>not initialized</td>
</tr>
<tr>
<td>NON-NULL</td>
<td>it holds a reference to an internal object in the VPE DLL</td>
</tr>
</tbody>
</table>

**See also:**

dycodoc Template Processing in the Programmer's Manual
29.3 IsNull

Tests whether a Field has a null-value. A field has a null-value after calling ClearFields or after explicitly assigning a null-value to a Field by either calling TVPEField.SetNull or TVPETemplate.SetFieldToNull.

**property boolean TVPEField.IsNull**

read; runtime only

**Remarks:**

In case of an error, an exception is thrown (for example out of memory).

The [Template Object](#) itself offers simple methods to access a Field's value directly, i.e. without retrieving a Field Object first. See GetFieldIsNull and SetFieldToNull.

**See also:**

dycodoc Template Processing in the Programmer's Manual
29.4 SetNull
Assigns a null-value to a field. A field has a null-value after calling TVPETemplate.ClearFields or after explicitly assigning a null-value to a Field by either calling TVPEField.SetNull or by calling this method.

**method** void TVPEField.SetNull

**Remarks:**
The Template Object itself offers simple methods to access a Field's value directly, i.e. without retrieving a Field Object first. See GetFieldIsNull and SetFieldToNull.

**See also:**
dycodoc Template Processing in the Programmer's Manual
29.5 NullValueText

Sets / returns the text which is printed for Fields that have a null-value. You can assign a special text to Fields with a null-value. By default, the null-value text is an empty string. You can specify for each field an individual null-value text, for example "n/a".

**property string TVPEField.NullValueText**

read / write; runtime only

**Remarks:**
In case of an error, an exception is thrown (for example out of memory).

The Template Object itself offers simple methods to access a Field's value directly, i.e. without retrieving a Field Object first. See GetFieldNullValueText and SetFieldNullValueText.

**See also:**
dycodoc Template Processing in the Programmer's Manual
29.6 AsString
Sets / returns the value of a Field as string.

property string TVPEField.AsString
  read / write; runtime only

Remarks:
In case of an error, an exception is thrown (for example out of memory).

The Template Object itself offers simple methods to access a Field's value directly, i.e. without retrieving a Field Object first. See GetFieldAsString and SetFieldAsString.

See also:
dycodoc Template Processing in the Programmer’s Manual
29.7 AsInteger
Sets / returns the value of a Field as integer.

property integer TVPEField.AsInteger
    read / write; runtime only

Remarks:
In case of an error, an exception is thrown (for example out of memory or the field contained
pure text that could not be converted to an integer value).

The Template Object itself offers simple methods to access a Field's value directly, i.e. without
retrieving a Field Object first. See GetFieldAsInteger and SetFieldAsInteger.

See also:
dycodoc Template Processing in the Programmer's Manual
29.8 AsNumber

Sets / returns the value of a Field as number (double).

property double TVPEField.AsNumber

   read / write; runtime only

Remarks:
In case of an error, an exception is thrown (for example out of memory or the field contained pure text that could not be converted to a number value).

The Template Object itself offers simple methods to access a Field's value directly, i.e. without retrieving a Field Object first. See GetFieldAsNumber and SetFieldAsNumber.

See also:
dycodoc Template Processing in the Programmer's Manual
29.9 AsBoolean
Sets / returns the value of a Field as boolean.

property Boolean [integer] TVPEField.AsBoolean

read / write; runtime only

Remarks:
In case of an error, an exception is thrown (for example out of memory or the field contained
pure text that could not be converted to an boolean value).

The Template Object itself offers simple methods to access a Field's value directly, i.e. without
retrieving a Field Object first. See GetFieldAsBoolean and SetFieldAsBoolean.

See also:
dycodoc Template Processing in the Programmer's Manual
29.10 SetAsDateTime

Sets / returns the value of a Field as date / time.

```cpp
method integer TVPEField.SetAsDateTime(
    integer year,
    integer month,
    integer day,
    integer hour,
    integer minute,
    integer second,
    integer msec
)
```

`int year, month, day, hour, minute, second, msec`

the integer values, which shall be assigned to the respective values of the Field as integer

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>if the field was found and its value could be set</td>
</tr>
<tr>
<td>False</td>
<td>otherwise</td>
</tr>
</tbody>
</table>

**Remarks:**

The Template Object itself offers simple methods to access a Field's value directly, i.e. without retrieving a Field Object first. See SetFieldAsDateTime.

**See also:**

dycodoc Template Processing in the Programmer's Manual
29.11 AsOleDateTime

[not in .NET]
Sets / returns the value of a Field as OLE date / time.

property date TVPEField.AsOleDateTime

    read / write; runtime only

Remarks:
In case of an error, an exception is thrown (for example out of memory or the field contained pure text that could not be converted to an OLE date / time value).

The Template Object itself offers simple methods to access a Field's value directly, i.e. without retrieving a Field Object first. See GetFieldAsOleDateTime and SetFieldAsOleDateTime.

See also:
dycodoc Template Processing in the Programmer's Manual
29.12 AsNetDateTime
Sets / returns the value of a Field as .NET date / time.

property DateTime TVPEField.AsNetDateTime
    read / write; runtime only

Remarks:
In case of an error, an exception is thrown (for example out of memory or the field contained pure text that could not be converted to a .NET date / time value).

The Template Object itself offers simple methods to access a Field’s value directly, i.e. without retrieving a Field Object first. See GetFieldAsNetDateTime and SetFieldAsNetDateTime.

See also:
dycodoc Template Processing in the Programmer's Manual
29.13 AsJavaDateTime

Sets / returns the value of a Field as Java date / time (number of milliseconds since midnight Jan 1, 1970).

**property double TVPEField.AsJavaDateTime**

read / write; runtime only

**Remarks:**
In case of an error, an exception is thrown (for example out of memory or the field contained pure text that could not be converted to a Java date / time value).

The Template Object itself offers simple methods to access a Field's value directly, i.e. without retrieving a Field Object first. See `GetFieldAsJavaDateTime` and `SetFieldAsJavaDateTime`.

**See also:**
dycodoc Template Processing in the Programmer's Manual
29.14 Name

Returns the name of the Field as defined in *dycodoc*.

**property string TVPEField.Name**

read; runtime only

**Returns:**

the name of the Field as defined in *dycodoc*

**See also:**

dycodoc Template Processing in the Programmer's Manual
29.15 Description

Returns the description of the Field as defined in *dycodoc*.

**property string TVPEField.Description**

read; runtime only

**Returns:**

the description of the Field as defined in *dycodoc*

**See also:**

dycodoc Template Processing in the Programmer's Manual
29.16 DataSourceObject
Returns the Data Source Object where the Field belongs to.

property TVPDataSource TVPEField.DataSourceObject
   read; runtime only

Returns:
the Data Source Object where the Field belongs to

Remarks:
The function can be used to access the Field's Data Source Object directly.

See also:
dycodoc Template Processing in the Programmer's Manual
Important Note for VPE-VCL Users in the Programmer's Manual
30 Template Page Object

30.1 Template Page Object - TVPETemplatePage

[Enterprise Edition and above]

This section describes the methods and properties of the Page Object.
The class-name is TVPETemplatePage.

.NET: The TVPETemplatePage class is derived from the base class Object.

See also:

dycodoc Template Processing in the Programmer's Manual
30.2 ObjectHandle

This is the internal object handle of the VPE DLL for the TemplatePage Object.
The internal Object Handle is of interest to check if its value is null, which means that it is not initialized, i.e. it has no corresponding object in the DLL. This can happen in case of errors, i.e. if the object is not existing.

**property long TVPETemplatePage.ObjectHandle**

read; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>not initialized</td>
</tr>
<tr>
<td>NON-NULL</td>
<td>it holds a reference to an internal object in the VPE DLL</td>
</tr>
</tbody>
</table>

**See also:**

*dydoc* Template Processing in the Programmer's Manual
30.3 PageWidth
Sets / returns the width of the specified template page.

property VpeCoord TVPETemplatePage.PageWidth
  read / write; runtime only

See also:
  dycodoc Template Processing in the Programmer’s Manual
30.4 PageHeight
Sets / returns the height of the specified template page.

property VpeCoord TVPETemplatePage.PageHeight
   read / write; runtime only

See also:
   dycodoc Template Processing in the Programmer's Manual
30.5 Orientation
Sets / returns the orientation of the specified template page.

   read / write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VORIENT_PORTRAIT</td>
<td>1</td>
<td>Portrait</td>
<td></td>
</tr>
<tr>
<td>VORIENT_LANDSCAPE</td>
<td>2</td>
<td>Landscape</td>
<td></td>
</tr>
</tbody>
</table>

See also:
dycodoc Template Processing in the Programmer's Manual
### 30.6 PaperBin

Sets / returns the printer's input paper bin of the specified template page.


read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBIN_UNTOUCHED</td>
<td>0</td>
<td>Untouched</td>
<td></td>
</tr>
<tr>
<td>VBIN_UPPER</td>
<td>1</td>
<td>Upper</td>
<td></td>
</tr>
<tr>
<td>VBIN_ONLYONE</td>
<td>1</td>
<td>OnlyOne</td>
<td></td>
</tr>
<tr>
<td>VBIN_LOWER</td>
<td>2</td>
<td>Lower</td>
<td></td>
</tr>
<tr>
<td>VBIN_MIDDLE</td>
<td>3</td>
<td>Middle</td>
<td></td>
</tr>
<tr>
<td>VBIN_MANUAL</td>
<td>4</td>
<td>Manual</td>
<td></td>
</tr>
<tr>
<td>VBIN_ENVELOPE</td>
<td>5</td>
<td>Envelope</td>
<td></td>
</tr>
<tr>
<td>VBIN_ENVMANUAL</td>
<td>6</td>
<td>EnvelopeManual</td>
<td></td>
</tr>
<tr>
<td>VBIN_AUTO</td>
<td>7</td>
<td>Auto</td>
<td></td>
</tr>
<tr>
<td>VBIN_TRACTOR</td>
<td>8</td>
<td>Tractor</td>
<td></td>
</tr>
<tr>
<td>VBIN_SMALLFMT</td>
<td>9</td>
<td>SmallFormat</td>
<td></td>
</tr>
<tr>
<td>VBIN_LARGEFMT</td>
<td>10</td>
<td>LargeFormat</td>
<td></td>
</tr>
<tr>
<td>VBIN_LARGECAPACITY</td>
<td>11</td>
<td>LargeCapacity</td>
<td></td>
</tr>
<tr>
<td>VBIN_CASSETTE</td>
<td>14</td>
<td>Cassette</td>
<td></td>
</tr>
</tbody>
</table>

Not all of the bin options are available on every printer. Check the printer's documentation for more specific descriptions of these options.

**See also:**

dycodoc Template Processing in the Programmer's Manual
30.7 nLeftMargin, nTopMargin, nRightMargin, nBottomMargin

Sets / returns the position of the left, top, right and bottom margins of the specified template page. See "Dynamic Positioning" and "Page Margins" in the Programmer's Manual for details.

```plaintext
property VpeCoord TVPETemplatePage.nLeftMargin
property VpeCoord TVPETemplatePage.nTopMargin
property VpeCoord TVPETemplatePage.nRightMargin
property VpeCoord TVPETemplatePage.nBottomMargin

read / write; runtime only
```

See also:

dycodoc Template Processing in the Programmer's Manual
30.8 VpeObjectCount

Returns the number of VPE Objects in the template page.

property long TVPETemplatePage.VpeObjectCount

read; runtime only

Returns:
the number of VPE Objects in the template page

See also:
dycodoc Template Processing in the Programmer's Manual
30.9 VpeObject

Returns a VPE Object from a template page.

method TVPEObject TVPETemplatePage.VpeObject(
   long Index
)

long Index
   index into the array of available VPE Objects in the page
   this value must be in the range between 0 and VpeObjectCount() - 1

Returns:
a VPE Object from a template page

Remarks:
The function can be used to access VPE Objects within the template directly.

See also:
dycodoc Template Processing in the Programmer's Manual
Important Note for VPE-VCL Users in the Programmer's Manual
31 VPE Object

31.1 VPE Object - TVPEObject

[Enterprise Edition and above]
This section describes the methods and properties of the VPE Object.
The class-name is TVPEObject.

.NET: The TVPEObject class is derived from the base class Object.

In addition the VPE Object has the following properties and methods:

- PenSize, PenStyle, PenColor, SetPen, NoPen
- BkgMode, TransparentMode, BkgColor, BkgGradientStartColor, BkgGradientEndColor, BkgGradientTriColor, BkgGradientMiddleColorPosition, BkgGradientMiddleColor, BkgGradientRotation, HatchStyle, HatchColor, CornerRadius
- FontName, FontSize, CharSet, TextAlignment, TextBold, TextUnderline, TextUnderlined, TextStrikeOut, TextItalic, TextColor, SelectFont, SetFont, SetFontAttr
- Rotation
- PictureType, PicturePage, PictureEmbedInDoc, PictureKeepAspect, PictureBestFit, PictureX2YResolution, PictureDrawExact
- SetBarcodeParms, BarcodeAlignment, BarcodeMainTextParms, BarcodeAddTextParms, BarcodeAutoChecksum, BarcodeThinBar, BarcodeThickBar
- Bar2DAlignment
- DataMatrixEncodingFormat
- DataMatrixEccType
- DataMatrixRows
- DataMatrixColumns
- DataMatrixMirror
- DataMatrixBorder
- QRCODEVersion
- QRCODEEccLevel
- QRCODEMode
- QRCODEBorder
- PDF417ErrorLevel
- PDF417Rows
- PDF417Columns
- AztecFlags
- AztecControl
- AztecMenu
AztecMultipleSymbols
AztecID

**Viewable, Printable, Streamable, Shadowed**

UDOIParam

CharCount, DividerPenSize, DividerPenColor, AltDividerNPosition, AltDividerPenSize, AltDividerPenColor, BottomLinePenSize, BottomLinePenColor, DividerStyle, AltDividerStyle, FormFieldFlags

CheckmarkColor

**See also:**

dycodoc Template Processing in the Programmer's Manual
31.2 ObjectHandle

This is the internal object handle of the VPE DLL for the VPE Object.

The internal Object Handle is of interest to check if its value is null, which means that it is not initialized, i.e. it has no corresponding object in the DLL. This can happen in case of errors, i.e. if the object is not existing.

**property long TVPEObject.ObjectHandle**

read; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>not initialized</td>
</tr>
<tr>
<td>NON-NULL</td>
<td>it holds a reference to an internal object in the VPE DLL</td>
</tr>
</tbody>
</table>

**See also:**

dycodoc Template Processing in the Programmer's Manual
31.3 Kind
By using this function, you can identify the type of an object.

property Kind [long] TVPEObject.Kind
read; runtime only

Returns:
Possible return codes are:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOBJID_NULL</td>
<td>16</td>
<td>Null</td>
<td>does not occur</td>
</tr>
<tr>
<td>VOBJID_LINE</td>
<td>17</td>
<td>Line</td>
<td></td>
</tr>
<tr>
<td>VOBJID_POLYLIME</td>
<td>18</td>
<td>Polyline</td>
<td></td>
</tr>
<tr>
<td>VOBJID_FRAME</td>
<td>19</td>
<td>Frame</td>
<td></td>
</tr>
<tr>
<td>VOBJID_TEXT</td>
<td>20</td>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>VOBJID_PICTURE</td>
<td>21</td>
<td>Picture</td>
<td></td>
</tr>
<tr>
<td>VOBJID_BARCODE</td>
<td>22</td>
<td>Barcode</td>
<td></td>
</tr>
<tr>
<td>VOBJID_ELLIPSE</td>
<td>23</td>
<td>Ellipse</td>
<td></td>
</tr>
<tr>
<td>VOBJID_PIE</td>
<td>24</td>
<td>Pie</td>
<td></td>
</tr>
<tr>
<td>VOBJID_POLYGON</td>
<td>25</td>
<td>Polygon</td>
<td></td>
</tr>
<tr>
<td>VOBJID_RTF</td>
<td>26</td>
<td>RTF</td>
<td></td>
</tr>
<tr>
<td>VOBJID_CHART</td>
<td>27</td>
<td>Chart</td>
<td></td>
</tr>
<tr>
<td>VOBJID_UDO</td>
<td>28</td>
<td>UDO</td>
<td></td>
</tr>
<tr>
<td>VOBJID_FORMFIELD</td>
<td>29</td>
<td>FormField</td>
<td></td>
</tr>
<tr>
<td>VOBJID_RESERVED</td>
<td>30</td>
<td>----------</td>
<td>does not occur</td>
</tr>
<tr>
<td>VOBJID_DOCDATA</td>
<td>31</td>
<td>DocData</td>
<td>VPE document handle</td>
</tr>
<tr>
<td>VOBJID_CTRL_FORMFIELD</td>
<td>32</td>
<td>CtrlFormField</td>
<td></td>
</tr>
<tr>
<td>VOBJID_CTRL_CHECKBOX</td>
<td>33</td>
<td>CtrlCheckbox</td>
<td></td>
</tr>
<tr>
<td>VOBJID_CTRL_RADIOBUTTON</td>
<td>34</td>
<td>CtrlRadioButton</td>
<td></td>
</tr>
<tr>
<td>VOBJID_CTRL_RADIOBUTTONGROUP</td>
<td>35</td>
<td>CtrlRadioButtonGroup</td>
<td></td>
</tr>
<tr>
<td>VOBJID_RESERVED2</td>
<td>36</td>
<td>----------</td>
<td>does not occur</td>
</tr>
<tr>
<td>VOBJID_RESERVED3</td>
<td>37</td>
<td>----------</td>
<td>does not occur</td>
</tr>
<tr>
<td>VOBJID_DATA_MATRIX</td>
<td>38</td>
<td>DataMatrix</td>
<td></td>
</tr>
<tr>
<td>VOBJID_MAXI_CODE</td>
<td>39</td>
<td>MaxiCode</td>
<td></td>
</tr>
<tr>
<td>VOBJID_PDF417</td>
<td>40</td>
<td>Pdf417</td>
<td></td>
</tr>
<tr>
<td>VOBJID_AZTEC</td>
<td>41</td>
<td>Aztec</td>
<td></td>
</tr>
<tr>
<td>VOBJID_QRCODE</td>
<td>42</td>
<td>QRCODE</td>
<td></td>
</tr>
</tbody>
</table>
Example:

ActiveX - VB 6:

' Returns the type name of a given VPE Object-ID:
Private Function GetObjectKindName(obj As TVPEObject) As String
    Select Case obj.Kind
        Case VOBJID_LINE
            GetObjectKindName = "Line"
        Case VOBJID_POLYLINE
            GetObjectKindName = "Polyline"
        Case VOBJID_FRAME
            GetObjectKindName = "Frame"
        Case VOBJID_TEXT
            GetObjectKindName = "Text"
        Case VOBJID_PICTURE
            GetObjectKindName = "Picture"
        Case VOBJID_BARCODE
            GetObjectKindName = "Barcode"
        Case VOBJID_ELLIPSE
            GetObjectKindName = "Ellipse"
        Case VOBJID_PIE
            GetObjectKindName = "Pie"
        Case VOBJID_POLYGON
            GetObjectKindName = "Polygon"
        Case VOBJID_RTF
            GetObjectKindName = "RTF"
        Case VOBJID_CHART
            GetObjectKindName = "Chart"
        Case VOBJIDUDO
            GetObjectKindName = "UDO"
        Case VOBJID_FORMFIELD
            GetObjectKindName = "FormField"
        Case VOBJID_DOCDATA
            GetObjectKindName = "Document Data"
        Case VOBJID_CTRL_FORMFIELD
            GetObjectKindName = "FormField Control"
        Case VOBJID_CTRL_CHECKBOX
            GetObjectKindName = "Checkbox Control"
        Case VOBJID_CTRL_RADIOBUTTON
            GetObjectKindName = "RadioButton Control"
        Case VOBJID_CTRL_RADIOBUTTONGROUP
            GetObjectKindName = "RadioButton Group Control"
        Case VOBJID_DATA_MATRIX
            GetObjectKindName = "DataMatrix Barcode"
        Case VOBJID_MAXICODE
            GetObjectKindName = "MaxiCode Barcode"
        Case VOBJID_PDF417
            GetObjectKindName = "PDF417 Barcode"
        Case VOBJID_AZTEC
            GetObjectKindName = "Aztec Barcode"
        Case VOBJID_QRCODE
            GetObjectKindName = "QRCode Barcode"
        Case Else
            GetObjectKindName = "NULL"
    End Select
End Function

See also:

dycodoc Template Processing in the Programmer's Manual
31.4 Name

In dycodoc a unique name is assigned to each VPE Object. This function returns the name of a VPE Object as defined in dycodoc.

**property string TVPEObject.Name**

read; runtime only

**Remarks:**
This property does only return correct values for VPE Objects Field object and Control Objects that reside in a template, and which have been inserted into the document using DumpTemplate() / DumpTemplatePage(). Otherwise this property returns an empty string. You may access this property also for a VPE object that has a NULL-Handle, in this case an empty string is returned.

**See also:**
dycodoc Template Processing in the Programmer's Manual
31.5 Text
Returns the text content of a Text object.

**property string TVPEObject.Text**

read; runtime only

**Remarks:**
This function does not support RTF objects.

The returned string does not contain Fields nor the values of Fields.
See ResolvedText.

**See also:**
dycodoc Template Processing in the Programmer's Manual
31.6 ResolvedText

Returns the text content of a Text object. If Fields are used within the text, the current values of those Fields are merged into the text.

property string TVPEObject.ResolvedText

   read; runtime only

Remarks:
   This function does not support RTF objects.

See also:
   dycodoc Template Processing in the Programmer's Manual
31.7 nLeft, nTop, nRight, nBottom

Returns the coordinates, i.e. the position, of the specified VPE Object.

property long TVPEObject.nLeft
property long TVPEObject.nTop
property long TVPEObject.nRight
property long TVPEObject.nBottom

read; runtime only

Remarks:
The coordinate is normalized, i.e. the following rule is fulfilled:
  right >= left and bottom >= top
Please note that all VPE API methods (except VpeLine()) require normalized coordinates.

See also:
dycodoc Template Processing in the Programmer's Manual
31.8 PictureFileName

Sets / returns the file name of a Picture Object.

property string TVPEObject.PictureFileName

    read / write; runtime only

Remarks:
Writing to this property (i.e. assigning a value to it) does only work for Picture Objects that reside in a template. If you specify a different object, the function does nothing.

See also:
dycodoc Template Processing in the Programmer's Manual
31.9 TemplateObject
Returns the template object where the given object belongs to.

property TVPETemplate TVPEObject.TemplateObject
    read; runtime only

Remarks:
This property does only work for VPE Objects that reside in a template, for Field object and Control Objects. If you read this property from a different object, an exception is thrown.

See also:
dycodoc Template Processing in the Programmer's Manual
Important Note for VPE-VCL Users in the Programmer's Manual
31.10 InsertedVpeObjectCount

After a call to DumpTemplate() or DumpTemplatePage() the VPE Objects of a template are dumped into the VPE Document. Depending on the amount of text a VPE Object contains, it might have been split over multiple pages. When an object is split over multiple pages, each splitted object is a new distinct object. This function returns the number of VPE Objects in the document that have been created from the provided template VPE Object.

**property long TVPEObject.InsertedVpeObjectCount**

- read; runtime only

**Returns:**
- the number of VPE Objects in the document that have been created from the provided template VPE Object

**Remarks:**
- This property does only work for VPE Objects that reside in a template.

**See also:**
- dycodoc Template Processing in the Programmer's Manual
31.11 InsertedVpeObject

Returns the n-th VPE Object in the document that has been created from the provided template VPE Object.

**method** 

TVPO

et TVPEObject TVPEObject.InsertedVpeObject ( 

long Index 

) 

*long Index*

index into the array of available VPE objects in the VPE Document

this value must be in the range between 0 and InsertedVpeObjectCount - 1

**Returns:**

The n-th VPE Object in the document that has been created from the provided template VPE Object. If Index is out of range an exception is thrown.

**Remarks:**

This method does only work for VPE Objects that reside in a template.

**See also:**

dycodoc Template Processing in the Programmer's Manual

Important Note for VPE-VCL Users in the Programmer's Manual
### 31.12 InsertedVpeObjectPageNo

Returns the page number where the n-th VPE Object - that has been created from the provided template VPE Object - has been inserted into the document.

```csharp
method long TVPEObject.InsertedVpeObjectPageNo(
    long Index
)
```

*Index*
index into the array of available VPE objects in the VPE Document
this value must be in the range between 0 and InsertedVpeObjectCount - 1

**Returns:**
the page number where the n-th VPE Object - that has been created from the provided template VPE Object - has been inserted into the document

**Remarks:**
This method does only work for VPE Objects that reside in a template.

**See also:**
dycodoc Template Processing in the Programmer's Manual
31.13 **NextObject**

Returns the succeeding VPE Object in the document.

**property TVPEObject TVPEObject.NextObject**

read; runtime only

**Returns:**

The succeeding VPE Object in the document. If there is no succeeding object the `ObjectHandle` of the returned object is null.

**Remarks:**

This property allows in conjunction with the method `FirstObject` to iterate through all objects of a page / document.

This function does only work for VPE Objects that reside in a VPE Document.

**See also:**

dycodoc Template Processing in the Programmer's Manual
Important Note for VPE-VCL Users in the Programmer's Manual
32 Interactive Objects

32.1 Interactive Objects

[Interactive Edition only]

This section describes the methods and properties related to Interactive Objects.

See also:

32.2 FormFieldControl

Inserts either an *Interactive FormField* or an *Interactive Text* control into the document. The value of the property **CharCount** decides, which type of control is inserted:

If CharCount is > 1: an *Interactive FormField* control is inserted, and CharCount determines the number of character cells.

If CharCount is = 0: an *Interactive Text* control is inserted, and an unlimited number of characters (until the visible portion of the control is filled-up) may be entered by the user into the control.

If CharCount is < 0: an *Interactive Text* control is inserted, and CharCount determines the maximum number of characters which may be entered by the user into the control.

In case an *Interactive FormField* control is inserted, the same rules apply as described for the **FormField**.

In case an *Interactive Text* control is inserted, the control is editable over multiple lines, instead of a single line as with the *Interactive FormField* control. The control can be filled up with as many characters as will fit into the given rectangle of the object. In addition, the text alignment of *Interactive Text* control may be chosen freely (left, right, centered, justified).

**method TVPEObject VPE.FormFieldControl**

```
VpeCoord Left,
VpeCoord Top,
VpeCoord Right,
VpeCoord Bottom,
string Text
```

**VpeCoord Left, Top, Right, Bottom**

position and dimensions

**string Text**

the editable text displayed in the control (the control works with an internal copy of this text, it does not write to the memory pointed at by **text**)

**Returns:**

The **VPE Object** which represents the control. This VPE Object can be used later in your code to, for example, set the input focus on it or to retrieve its value and to change some of its properties.

**Remarks:**

VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition **Text**, **Rich Text** and **Picture** objects are able to compute their dimensions automatically depending on their visual content. For details please see “Dynamic Positioning” in the Programmer's Manual.

If you are using a multi-line Interactive Text control, lines are separated with the characters #13 + #10. (ASCII Code 13 plus ASCII Code 10)
Example:

```vba
// Insert an Interactive Text Control, where an unlimited number of
// characters may be entered:
Dim Control1 as TVPEObject
Dim Control2 as TVPEObject
Dim Control3 as TVPEObject
Doc.CharCount = 0
Control1 = Doc.FormFieldControl(1, 1, -9, VFREE,
   "Hello" + #10 + #13 + "World!"
)

// Insert an Interactive Text Control, limited to max. 14 characters:
Doc.CharCount = -14
Control2 = Doc.FormFieldControl(1, 2, -9, VFREE,
   "Hello" + #10 + #13 + "World!"
)

// Insert an Interactive Form Field Control, limited to max. 14 characters:
Doc.CharCount = 14
Control3 = Doc.FormFieldControl(1, 3, -9, VFREE, "Hello World!")
```

.NET:

use [Doc.nFree](#) instead of VFREE in the example above

See also:

32.3 Checkbox

Inserts a Checkbox into the document.

```
method TVPEObject VPE.Checkbox(
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom
)
```

*VpeCoord Left, Top, Right, Bottom*

position and dimensions

**Returns:**
The VPE Object which represents the control. This VPE Object can be used later in your code to, for example, set the input focus on it or to retrieve its value and to change some of its properties.

**Remarks:**
VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

**Example:**
```
Dim Control as TVPEObject
Control = Doc.Checkbox(1, 1, -0.5, -0.5)
```

**See also:**
### 32.4 RadioButtonGroup

Inserts a Radiobutton Group into the document. It is required that you insert a Radiobutton Group into the document before inserting Radiobuttons. A Radiobutton Group keeps the associated Radiobuttons together, controls the state switching if a button of the group is clicked and holds the value of the group.

**method TVPEObject VPE.RadioButtonGroup(**

  integer DefaultValue

**)

*integer DefaultValue*
the default value of the group

E.g. if a Radiobutton Group has three associated Radiobuttons, say button A with the value 1, button B with the value 2 and button C with the value 3 and you set default_value = 2, then the button B will be checked automatically when it is created and associated with this Group.

**Returns:**

The VPE Object which represents the control. The Object is required to associate Radiobuttons with it. In addition the VPE Object can be used later in your code to, for example, set the input focus on it or to retrieve its value and to change some of its properties.

**Remarks:**

If you assign a value to a group, which is not defined by any Radiobutton, no Radiobutton is selected, i.e. the visual state of the group is undefined. This feature can be used to reflect NULL values, for example from databases.

**Example:**

see RadioButton

**See also:**

32.5 RadioButton

Inserts a Radiobutton into the document. Before inserting a Radiobutton, you need to insert a Radiobutton Group into the document.

method TVPEObject VPE.RadioButton(
    TVPEObject RadioButtonGroup,
    VpeCoord Left,
    VpeCoord Top,
    VpeCoord Right,
    VpeCoord Bottom,
    integer Value
)

TVPEObject RadioButtonGroup
    the Radiobutton Group Object you want to associate the Radiobutton with

VpeCoord Left, Top, Right, Bottom
    position and dimensions

integer Value
    the value the Radiobutton shall represent within its group

Returns:
    The VPE Object which represents the control. This VPE Object can be used later in your code to, for example, set the input focus on it or to retrieve its value and to change some of its properties.

Remarks:
    VPE offers several methods to attach an object's position to margins and relative to the position of previously inserted objects. In addition Text, Rich Text and Picture objects are able to compute their dimensions automatically depending on their visual content. For details please see "Dynamic Positioning" in the Programmer's Manual.

Example:

```vpe
// Create a Radiobutton Group, where the button which represents // the value "2" shall be initially selected
Dim GroupObj as TVPEObject
GroupObj = Doc.RadioButtonGroup(2)

// Create three Radiobuttons and associate each with the group
Doc.RadioButton(GroupObj, 1, 1, -0.5, -0.5, 1)
Doc.Print(Doc.nRight + 15, VTOP, "lives in forrest")

Doc.RadioButton(GroupObj, VLEFT, Doc.nBottom + 0.5, -0.5, -0.5, 2)
Doc.Print(Doc.nRight + 15, VTOP, "lives on boat")

Doc.RadioButton(GroupObj, VLEFT, Doc.nBottom + 0.5, -0.5, -0.5, 3)
Doc.Print(Doc.nRight + 15, VTOP, "lives in house")

.NET:
use Doc.nTop and Doc.nLeft instead of VTOP and VLEFT in the example above
```

See also:

32.6 CheckmarkColor
Sets the checkmark color used for Checkboxes and Radiobuttons.

property Color VPE.CheckmarkColor
read / write; runtime only

Possible Values:
any of the "COLOR_xyz" constants described in Programmer's Manual
or ActiveX / VCL: any RGB value
or .NET: any member of the .NET Color structure

Default:
COLOR_BLACK

See also:
32.7 DocContainsControls
Determines, whether the document contains controls (i.e. Interactive Objects) or not.

**property boolean VPE.ContainsControls**
read; runtime only

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>yes, the document contains controls</td>
</tr>
<tr>
<td>False</td>
<td>no</td>
</tr>
</tbody>
</table>

**See also:**
32.8 ControlsModified

VPE keeps track, if the content (or value) of any control in the document has been modified. This property allows to read and explicitly set or reset the modification state of the document.

**property boolean VPE.ControlsModified**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>the document is modified</td>
</tr>
<tr>
<td>False</td>
<td>the document is not modified</td>
</tr>
</tbody>
</table>

**See also:**

32.9 Interaction

By default, a document which contains controls is not automatically editable by the user, i.e. all controls are locked for editing.

Set this property in order to enable or disable the whole document for editing.

A disabled control can not receive the focus, therefore the user can not change the control's value, neither with the keyboard, nor with the mouse. It is still possible to change the values of disabled controls by code.

The interaction for a document can not be enabled, if the document is stream based, i.e. it is opened using `SwapFileName`. Stream based documents are not editable. If you want the user to edit a .VPE document file, open a memory based document (i.e. use `OpenDoc()`) and read the document file into memory using `ReadDoc()`.

**property boolean VPE.Interaction**

*read / write; runtime only*

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>enable the whole document for editing</td>
</tr>
<tr>
<td>False</td>
<td>disable the whole document for editing</td>
</tr>
</tbody>
</table>

**Default:**

False

**See also:**

32.10 SetFocusToFirstControl

The focus is set to the very first control in the document (i.e. the control with the lowest TAB-ID).

**method boolean VPE.SetFocusToFirstControl( )**

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success, the first control in the Tab-Order received the focus</td>
</tr>
<tr>
<td>False</td>
<td>otherwise, for example if every object - or the whole document - is disabled, or the object which currently owns the focus did not allow to remove the focus</td>
</tr>
</tbody>
</table>

**See also:**

### 32.11 SetFocusControlByName

This method can only be used, if a template containing controls (i.e. Interactive Objects) has been dumped into the VPE Document. In dycodoc, you can assign a name to each object. The name can be supplied to this method in order to set the focus to a named control.

```csharp
method boolean VPE.SetFocusControlByName(
    TVPETemplate objTemplate,
    string Name
)
```

**TVPETemplate objTemplate**
- Template Object of the template which was dumped into the VPE Document

**string Name**
- the name of the control as assigned in dycodoc

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success, the control was found and the focus could be set</td>
</tr>
<tr>
<td>False</td>
<td>otherwise, for example if every object - or the whole document - is disabled, or the object which currently owns the focus did not allow to remove the focus</td>
</tr>
</tbody>
</table>

**Remarks:**
If the parameter name is NULL or empty (""), the focus is removed from the control currently having the focus.

**Example:**

```csharp
// Set the focus on the control named "Street" in dycodoc:
Doc.SetFocusControlByObjName(objTemplate, "Street")
```

**See also:**
### 32.12 SetFocusControl

Sets the focus to a control.

**method boolean VPE.SetFocusControl(  
  TVPEObject objControl
)
**

**TVPEObject objControl**  
Control Handle - the focus is set to the control identified by this handle

**Returns:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>success, the control was found and the focus could be set</td>
</tr>
<tr>
<td>False</td>
<td>otherwise, for example if every object - or the whole document - is disabled, or the object which currently owns the focus did not allow to remove the focus</td>
</tr>
</tbody>
</table>

**Remarks:**
If the parameter *hControl* is NULL, the focus is removed from the control currently having the focus.

**Example:**

```vba
Dim objControl As TVPEObject
objControl = Doc.FormFieldControl(1, 1, -9, VFREE, "Hello World!")
Doc.SetFocusControl(objControl);
```

**.NET:**  
use [Doc.nFree](#) instead of VFREE in the example above

**See also:**  
32.13 GetFocusControl

Returns the VPE Object which currently owns the focus.

```csharp
method long VPE.GetFocusControl()
```

**Returns:**

Returns the Control Handle of the control which currently has the focus.
If no control owns the focus, the ObjectHandle of the returned object is null.

**Example:**

```csharp
// The following code retrieves the name of the focused control
// (assuming the control was created from a dycodoc template,
// otherwise controls don't have names assigned to them and an empty string
// is returned)

Dim control_name As String
control_name = Doc.GetFocusControl.Name
```

**See also:**

Interactive Documents in the Programmer’s Manual.
32.14 FindControl

Searches by name the VPE Object (a control) which was inserted with DumpTemplate() / DumpTemplatePage() into the current document. Names can only assigned to objects using dycodoc and templates.

This method belongs to the TVPETemplate Object!

Use a Template Object of a template which has already been dumped into the VPE Document.

```csharp
method TVPEObject VPE.FindControl(string Name)

string Name
the name of the control as assigned in dycodoc

Returns:
Returns the VPE Object of the control with the given name.
If no control with the given name was found, the ObjectHandle of the returned object is null.

Example:

// Retrieves the Control Handle of the object named "Street"
// and disables the control
objTemplate.FindControl("Street").ControlEnabled = False
```

See also:
32.15 ControlEnabled

Enables / disables the specified control. A disabled control can not receive the focus, therefore the user can not change the control's value, neither with the keyboard, nor with the mouse. It is still possible to change the values of disabled controls by code.

This method belongs to TVPEObject.

**property boolean VPE.ControlEnabled**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>the control is enabled for editing</td>
</tr>
<tr>
<td>False</td>
<td>the control is disabled for editing</td>
</tr>
</tbody>
</table>

**Remarks:**

You may access this property also for a VPE object that has a NULL-Handle, in this case a write access does nothing and a read access returns False.

**Example:**

```vbnet
// Retrieves the Control Handle of the object named "Street"
// and enables the control
objTemplate.FindControl("Street").ControlEnabled = True
```

**See also:**

32.16 ControlTabIndex

Returns / modifies the Tab-Index of the specified control. Each control has assigned a unique Tab-Index. The Tab-Index determines the order in which the user can Tab through the controls: if the user presses the Tab key on the keyboard, VPE will search the whole document for the control with the next higher Tab-Index and set the focus on it. The reverse is done, if the user presses Shift + Tab ("Backtab").

If you are using templates, the Tab-Index can be assigned to each object using dycodoc (in the object's properties dialog). If you are creating Controls by code (e.g. by calling `FormFieldControl()`, `Checkbox()`, etc.), VPE assigns automatically a Tab-Index to each object in the order of creation.

You can use this property to modify the Tab-Order of a document. But special care must be taken: make sure that you don't assign one and the same Tab-Index to more than one object. The Tab-Index must be unique for each object in the document.

This property belongs to TVPEObject.

**property long VPE.ControlTabIndex**

read / write; runtime only

**Possible Values:**

The Tab-Index of the specified control. If this property is used for a TVPEObject which does not represent a control, it will return null für reading and it will do nothing if you assign a value to it.

**Remarks:**

The values 0 and -1 (0xffffffff) are reserved. You can not set the Tab-Index to either value, in such case the original Tab-Index remains unchanged.

You may access this property also for a VPE object that has a NULL-Handle, in this case a write access does nothing and a read access returns zero.

**See also:**

32.17 ControlGroupObject

Returns the Group Object at which the specified control belongs to.
This property belongs to TVPEObject.

**property TVPEObject VPE.ControlGroupObject**

- **read; runtime only**

**Returns:**
- The Group Object the specified control belongs to.
- If the specified control does not belong to a group, or if this property is used for a TVPEObject which does not represent a control, the ObjectHandle of the returned object is null.

**Remarks:**
- You may access this property also for a VPE object that has a NULL-Handle, in this case an object with a NULL-Handle is returned.

**See also:**
32.18 ControlFieldObject

If the specified control was created from a template using dycodoc and \texttt{DumpTemplate()}, its value might be associated with a field. In such case this property returns the associated Field object.

This property belongs to TVPEObject.

\begin{verbatim}
property TVPEField VPE.ControlFieldObject
   read; runtime only

Returns:
The Field Object which is associated with the specified control.
If the specified control is not associated with a Field, or if this property is used for a TVPEObject which does not represent a control, the ObjectHandle of the returned object is null.

Remarks:
You may access this property also for a VPE object that has a NULL-Handle, in this case an object with a NULL-Handle is returned.
\end{verbatim}

See also:
Interactive Documents in the Programmer’s Manual.
32.19 ControlTplVpeObject

If the specified control in the document was created from a template using dycodoc and DumpTemplate(), you can retrieve the corresponding VPE Object of the Template (i.e. this is the VPE Object from which the Control was created) using this property.

This property belongs to TVPEObject.

property TVPEObject VPE.ControlTplVpeObject

   read; runtime only

Returns:
The corresponding VPE Object in the template from which the specified object in the document was created. If the specified object was not created from a template, the ObjectHandle of the returned object is null.

Remarks:
The property can be used to access VPE Objects within the template directly. Normally, there is no need to use this property.

You may access this property also for a VPE object that has a NULL-Handle, in this case an object with a NULL-Handle is returned.

See also:
32.20 ControlAsString

Returns / sets the value of the specified control as string.

This property belongs to TVPEObject.

**property string VPE.ControlAsString**

read / write; runtime only

**Remarks:**

If the control was created from a template and it has an associated field, you should not use this property. It is - from the programmer’s view and source code logic - more efficient to use instead the field’s methods and properties in order to access the control’s value. The corresponding property of the field is **AsString**.

In case of an error, an exception is thrown (for example, out of memory).

**Example:**

```vba
Dim RecipientCompany As String
Dim objControl As TVPEObject
objControl = Doc.FormFieldControl(1, 1, -9, VFREE, "Hello World!")
RecipientCompany = objControl.ControlAsString
objControl.ControlAsString = "IDEAL Software"
```

**.NET:**

use [Doc.nFree](#) instead of VFREE in the example above

**See also:**

Interactive Documents in the Programmer’s Manual.
32.21 ControlAsInteger

Returns / sets the value of the specified control as integer.
This property belongs to TVPEObject.

**property string VPE.ControlAsInteger**

read / write; runtime only

**Remarks:**
If the control was created from a template and it has an associated field, you should not use this property. It is - from the programmer's view and source code logic - more efficient to use instead the field's methods and properties in order to access the control's value. The corresponding property of the field is `AsInteger`.

In case of an error, an exception is thrown (for example out of memory or the field contained pure text that could not be converted to an integer value).

**Example:**

```vbnet
Dim value As Integer
Dim objControl As TVPEObject
objControl = Doc.FormFieldControl(1, 1, -9, VFREE, "123")
value = objControl.ControlAsInteger
objControl.ControlAsInteger = 456
```

**.NET:**
use `Doc.nFree` instead of `VFREE` in the example above

**See also:**
33 PDF Export

33.1 PDF Export

This section describes the methods and properties related to PDF Export.
A VPE Document is exported to the Adobe PDF file format by calling the method `WriteDoc()`.

For important details about PDF Export and the features provided by VPE, please see the Programmer's Manual, chapter "The PDF Export Module".
33.2 PDFVersion
Sets / returns the current version compatibility for exported PDF documents.

property PDFVersion [long] VPE.PDFVersion

read / write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPE_PDF_ACROBAT_4</td>
<td>1300</td>
<td>Acrobat4</td>
<td>PDF Version 1.3</td>
</tr>
<tr>
<td>VPE_PDF_ACROBAT_5</td>
<td>1400</td>
<td>Acrobat5</td>
<td>PDF Version 1.4</td>
</tr>
</tbody>
</table>

Default:
VPE_PDF_ACROBAT_4

Remarks:
The version compatibility is especially important for the encryption key length (see also Encryption). When you change this property, the encryption key length is set automatically to 40-bits for VPE_PDF_ACROBAT_4 and to 128-bits for VPE_PDF_ACROBAT_5.

Huge documents with Adobe Acrobat 4.0:
When viewing documents with a large number of pages, all pages after page number 34465 are not displayed and an error is reported. The critical page number may alter, depending on the number of other objects in the document.
In order to have documents correctly displayed in Adobe Acrobat 4.0, always keep the number of pages below 8000.
33.3 Author

property string VPE.Author

   write; runtime only

Default:
   not set (i.e. empty)
33.4 Title

Sets the Title of the exported document. This value is displayed in Acrobat Reader's Document Properties dialog box. If you do not set this property, VPE sets the title of the PDF document by default to the Caption property.

property string VPE.Title

    write; runtime only

Default:
    the value of the property Caption
33.5 Subject
Sets the Subject of the document. This value is displayed in Acrobat Reader's Document Properties dialog box.

**property string VPE.Subject**

write; runtime only

**Default:**
not set (i.e. empty)
33.6 Keywords
Sets the Keywords for the document. This value is displayed in Acrobat Reader's Document Properties dialog box.

property string VPE.Keywords

    write; runtime only

Default:
    not set (i.e. empty)
33.7 Creator

Sets the Creator of the document. This should be the name of your application. The value is displayed in Acrobat Reader's Document Properties dialog box.

**property string VPE.Creator**

write; runtime only

**Default:**
not set (i.e. empty)
33.8 EmbedAllFonts
[Not supported by the Community Edition]

Sets / returns the current setting for font embedding. Embedded fonts are stored inside the PDF document. Some fonts can not be embedded, because they have a copy-protection bit set. In this case, VPE will not embed the font.

**property boolean VPE.EmbedAllFonts**

read / write; runtime only

**Possible values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>the True-Type fonts that are used in the document are embedded</td>
</tr>
<tr>
<td>False</td>
<td>the True-Type fonts that are used in the document are not embedded</td>
</tr>
</tbody>
</table>

**Default:**

False

**Remarks:**

This property controls the default behaviour of VPE. You can override this property individually for each font by using the method `SetFontControl()`.

VPE does only embed True-Type fonts, it will not embed PostScript fonts.

You can also subset embedded fonts. Subsetting means that VPE builds and embeds on-the-fly a new font, which does only contain the characters used in the current document. This creates in regular smaller fonts and therefore smaller PDF files. See `SubsetAllFonts` for details.
33.9 DocExportPictureResolution

Sets / returns the maximum allowed X and Y DPI resolution for images exported to non-VPE documents (like PDF, HTML, etc.).

Only images of higher resolutions are scaled down to the specified resolution, images that have already lower resolutions than the specified are not rescaled.

PDF export: this only applies to images that are managed internally as bitmaps. JPG image files are copied in their original form directly to PDF documents, so they are not rescaled.

The lower the image resolution, the smaller are the created images / documents. But especially for printing, too low image resolutions cause low quality printouts.

**property integer VPE.DocExportPictureResolution**

read / write; runtime only

**Possible values:**
the maximum allowed image resolution

**Default:**
300 DPI

**Remarks:**
If the created documents are mainly used for previewing on the screen - and not for printing - we recommend to use 96 DPI. For printing, a resolution of 300 DPI is recommended.

VPE can rescale images using different algorithms and qualities (see DocExportPictureQuality for details).

Non-Windows platforms: this option requires the Professional Edition or higher. For lower editions no down-scaling is performed.
33.10 DocExportPictureQuality

[VPE Professional Edition and above]

Sets / returns the current setting for the quality of images exported to non-VPE documents (like PDF, HTML, etc.).

**property DocExportPictureQuality [integer] VPE.DocExportPictureQuality**

read / write; runtime only

**Possible values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICEXP_QUALITY_NORMAL</td>
<td>0</td>
<td>Normal</td>
<td>embedded images will have normal quality [default]</td>
</tr>
<tr>
<td>PICEXP_QUALITY_HIGH</td>
<td>1</td>
<td>High</td>
<td>embedded images will have high quality (using Scale2Gray, only available in VPE Professional Edition or higher)</td>
</tr>
</tbody>
</table>

**Default:**
PICEXP_QUALITY_NORMAL

**Remarks:**
If an image is scaled to a lower resolution during export of the document (see DocExportPictureResolution), this can either be done using a standard algorithm for stretching a bitmap (PICEXP_QUALITY_NORMAL), or it can be done using the Scale2Gray technology (PICEXP_QUALITY_HIGH), which gives much more accurate results.
33.11 FastWebView

[VPE Professional Edition and above]

Sets / returns the current type for exported PDF documents. If the document is exported to PDF and Fast Web View is activated, the document is written as web-optimized PDF file. Adobe calls this 'Linearized PDF', an enhanced PDF file format especially for the Internet. It allows to view any given page on a client site as fast as possible without downloading the whole document from a server.

This feature is especially useful, if documents are created on a server and are transmitted to clients via a network. The client will be able to view the first page of the transmitted document - and to browse to specific pages in the document - as fast as possible, while the transmission is still in process.

Whilst non-optimized PDF documents are created by VPE in one single pass, where every page is written to the PDF file immediately, web-optimized documents are created internally in two passes: VPE can not write any part of the document, before it has analysed the complete document structure, because it needs to arrange the objects in the PDF document in a very special way according to the Adobe PDF specifications. In addition, some special tables (called "Hint Tables"), which contain information about the document structure, have to be created.

The disadvantage of the optimized mode is, that VPE needs more processor time and memory to perform the optimization. During the first pass, VPE keeps all objects of the document in memory and analyses the document structure. In a second pass VPE writes the document to the file. Depending on the number of simultaneous users and the overall workload of the server, this is no problem, if small documents with a few pages are created.

In case you require to create huge documents simultaneously for many users, you can instruct VPE to use a special swapping technique, so instead of using RAM memory, VPE will swap the information for all the objects in the document to a temporary file. This is done with a special technique and works very fast. (see UseTempFiles for details)

We recommend that you monitor the workload of the server. Depending on available free ram during a typical workload scenario, you should decide whether to use the swapping technology or not. Some tests also revealed that - depending on the scenario and contents of the document - the use of temporary files is significantly faster than using memory based document creation!

So our second recommendation is to measure the performance during a typical workload scenario with - and without - swapping, to decide what mode of operation to use.

Each temporary swap file is deleted automatically after a PDF document has been created.

**property boolean VPE.FastWebView**

read / write; runtime only

**Possible values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>create linearized PDF documents for Fast Web View</td>
</tr>
<tr>
<td>False</td>
<td>create normal documents</td>
</tr>
</tbody>
</table>

**Default:**
False
33.12 UseTempFiles

[VPE Professional Edition and above]

Sets / returns the current mode of operation, specifying whether temporary swap files are used or not. This property is only in effect, if Fast Web View is activated.

**property boolean VPE.UseTempFiles**

read / write; runtime only

**Possible values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>temporary swap files are used to generate linearized PDF documents</td>
</tr>
<tr>
<td>False</td>
<td>RAM memory is used to generate linearized PDF documents</td>
</tr>
</tbody>
</table>

**Default:**

False

**Remarks:**

The temporary swap files are created in the directory specified by the TMP or TEMP environment variable. If both variables are not set, or the specified directory does not exist, the current working directory is used.

Each temporary swap file is deleted automatically after a PDF document has been created.
33.13 SubsetAllFonts

[VPE Professional Edition and above]

Sets the current mode for font subsetting. Only fonts that are embedded, can be subsetted (see EmbedAllFonts).

Font Subsetting means, that VPE assembles on-the-fly a new font from the source font, that contains only the characters which are used in the document. A subsetted font is in regular much smaller than the original font, which results in significantly smaller documents.

**property boolean VPE.SubsetAllFonts**

read / write; runtime only

**Possible values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>the True-Type fonts that are used in the document are subsetted</td>
</tr>
<tr>
<td>False</td>
<td>the True-Type fonts that are used in the document are not subsetted</td>
</tr>
</tbody>
</table>

**Default:**

False

**Remarks:**

This property controls the default behaviour of VPE. You can override this property individually for each font by using the method SetFontControl().

VPE does only subset True-Type fonts, it will not subset PostScript fonts.

**Please note:** Font Subsetting does not work correctly for Adobe Acrobat 4, if you are using character sets which are different from ANSI_CHARSET. So subsetting of all western character sets is supported for Acrobat 4, too.

**Example:**

The following code activates font embedding, which is required for font-subsetting, and then activates font-subsetting.

```plaintext
VPE.EmbedAllFonts  = True
VPE.SubsetAllFonts = True
```
33.14 SetFontControl

[VPE Professional Edition and above]

Allows to set several parameters for an individual font. This function overrides the settings of EmbedAllFonts and SubsetAllFonts.

method void VPE.SetFontControl(
    string FontName,
    string SubstFontName,
    boolean DoEmbed,
    boolean DoSubset
)

string FontName
    the font you want to set the parameters for

string SubstFontName
    the font with which you want to substitute the original font with

boolean DoEmbed
    
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>if the font is a True-Type font, it will be embedded into the document</td>
</tr>
<tr>
<td>False</td>
<td>the font will not be embedded into the document</td>
</tr>
</tbody>
</table>

boolean DoSubset
    
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>if the font is a True-Type font, and it is embedded, it will be subsetted</td>
</tr>
<tr>
<td>False</td>
<td>the font will not be subsetted</td>
</tr>
</tbody>
</table>

Remarks:

You can call this method repeatedly for each font you want to set individual parameters for. The settings can be deleted by calling the method ResetFontControl().

Whilst the settings for font substitution of this method are only active during the export to PDF documents, there is a second method to substitute fonts SetFontSubstitution(), which is always active.

For important details about fonts and font handling, please see the Programmer's Manual, chapter "Programming Techniques", subchapter "Fonts and Font Handling".

Example:

VPE.SetFontControl("Arial", "Helvetica", False, False)
VPE.SetFontControl("Times New Roman", "Times", False, False)

Instructs VPE to substitute "Arial" with "Helvetica" and "Times New Roman" with "Times".

Since Helvetica as well as Times are PostScript fonts, VPE can neither embed nor subset them. But because both fonts are standard Base 14 PostScript fonts, they are supported by PDF Readers on any platform, so embedding (and therefore subsetting) is not required.
33.15 ResetFontControl

[VPE Professional Edition and above]

With each call to SetFontControl(), VPE adds internally a new font control structure to a list. ResetFontControl() clears the whole list, so all individual font control settings are deleted.

method void VPE.ResetFontControl()
33.16 Encryption
[VPE Professional Edition and above]
Activates or deactivates encryption for exported PDF documents.

property Encryption [integer] VPE.Encryption

read / write; runtime only

Possible values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOC_ENCRYPT_NONE</td>
<td>0</td>
<td>Normal</td>
<td>no encryption [default]</td>
</tr>
<tr>
<td>DOC_ENCRYPT_STREAM</td>
<td>1</td>
<td>Stream</td>
<td>use stream encryption (compatible to Acrobat 4 and higher)</td>
</tr>
</tbody>
</table>

Default:
DOC_ENCRYPT_NONE

Remarks:
See EncryptionKeyLength, UserPassword, OwnerPassword and Protection for details.

If one and the same VPE document is exported multiple times to encrypted PDF documents, the size of the created PDF files might differ around 10 - 20 bytes. This is normal, because some encrypted characters - for example in the creation date / time string - need to be converted into a special form with additional escape characters.
33.17 EncryptionKeyLength
[VPE Professional Edition and above]

Sets the encryption key length for exported PDF documents. For documents which are compatible
to Acrobat Reader 4, only a value of 40 is allowed. For documents which are compatible to
Acrobat Reader 5, a value between 40 and 128 is allowed. If you want to create documents which
are compatible to Acrobat Reader 5 only, we recommend to use 128-bit encryption.

property integer VPE.EncryptionKeyLength

read / write; runtime only

Possible values:
the key length in bits in the range from 40 to 128;
the value must be a multiple of 8 (e.g. 40, 48, 56, 64, ..., 128)

Default:
40 if PDFVersion is VPE_PDF_ACROBAT_4 (default)
128 if PDFVersion is VPE_PDF_ACROBAT_5

Remarks:
The User Password as well as the Owner Password are used to create the encryption key. To
gain safe encryption, at least one of either passwords should be specified - better both.

As you set the property PDFVersion, the encryption key length is adjusted accordingly and any
previously assigned value is overwritten.
33.18 UserPassword
[VPE Professional Edition and above]

Sets the user password for the document. In order to be able to open an encrypted document (see Encryption), either this password - or the owner password - must be entered in advance. The document can not be opened, if an incorrect password is supplied (see also OwnerPassword).

If the document does not have a user password, no password is requested; Acrobat Reader can simply open, decrypt, and display the document.

Whether additional operations are allowed on a decrypted document depends on if either the user password or the owner password (if any) was supplied when the document was opened and on any access restrictions that were specified when the document was created.

Opening the document with the correct user password (or opening a document that does not have a user password) allows additional operations to be performed according to the user access permissions specified in the document’s encryption dictionary (see Protection).

**property string VPE.UserPassword**

read / write; runtime only

**Possible values:**
A string which may contain any characters. Up to 32 characters of the password are used for generating the encryption key (see EncryptionKeyLength).

**Default:**
not set (i.e. empty)
33.19 OwnerPassword

[VPE Professional Edition and above]

Sets the Owner Password for the document. Opening the document with the correct owner password (assuming it is not the same as the user password) allows full (owner) access to the document. This unlimited access includes the ability to change the document’s passwords and access permissions (see also UserPassword and Protection).

**property string VPE.OwnerPassword**

read / write; runtime only

**Possible values:**

A string which may contain any characters. Up to 32 characters of the password are used for generating the encryption key (see EncryptionKeyLength).

**Default:**

not set (i.e. empty)
### 33.20 Protection

[VPE Professional Edition and above]

Sets the document protection flags. Document protection is only in effect, if encryption is activated.

**property Protection [long] VPE.Protection**

read / write; runtime only

**Possible values:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDF_ALLOW_NONE</td>
<td>0</td>
<td>PDFAllowNone</td>
<td>full protection, no flag is raised</td>
</tr>
<tr>
<td>PDF_ALLOW_PRINT</td>
<td>4</td>
<td>PDFAllowPrint</td>
<td>allow to print the document (see also the flag PDF_ALLOW_HIQ_PRINT)</td>
</tr>
<tr>
<td>PDF_ALLOW_MODIFY</td>
<td>8</td>
<td>PDFAllowModify</td>
<td>allow to modify the contents of the document by operations other than those controlled by the flags PDF_ALLOW_TA_IFF, PDF_ALLOW_FILL_IFF and PDF_ALLOW_ASSEMBLE</td>
</tr>
<tr>
<td>PDF_ALLOW_COPY</td>
<td>16</td>
<td>PDFAllowCopy</td>
<td>PDF v1.3: allow to copy or otherwise extract text and graphics from the document, including extracting text and graphics (in support of accessibility to disabled users or for other purposes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PDF v1.4: allow to copy or otherwise extract text and graphics from the document by operations other than that controlled by PDF_ALLOW_EXTRACT</td>
</tr>
<tr>
<td>PDF_ALLOW_TA_IFF</td>
<td>32</td>
<td>PDFAllowTaIff</td>
<td>allow to add or modify text annotations, fill in interactive form fields, and - if PDF_ALLOW_MODIFY is also used - create or modify interactive form fields (including signature fields)</td>
</tr>
<tr>
<td>PDF_ALLOW_FILL_IFF</td>
<td>256</td>
<td>PDFAllowFillIff</td>
<td>PDF v1.4 only: allow to fill in existing interactive form fields (including signature fields), even if PDF_ALLOW_TA_IFF is not used</td>
</tr>
<tr>
<td>PDF_ALLOW_EXTRACT</td>
<td>512</td>
<td>PDFAllowExtract</td>
<td>PDF v1.4 only: allow to extract text and graphics (in support of accessibility to disabled users or for other purposes)</td>
</tr>
<tr>
<td>PDF_ALLOW_ASSEMBLE</td>
<td>1024</td>
<td>PDFAllowAssemble</td>
<td>PDF v1.4 only: allow to assemble the document (insert, rotate, or delete pages and create bookmarks or thumbnail images), even if PDF_ALLOW_MODIFY is not used</td>
</tr>
<tr>
<td>Constant</td>
<td>Value</td>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PDF_ALLOW_HIQ_PRINT</td>
<td>2048</td>
<td>PDFAllowHiQualityPrint</td>
<td>PDF v1.4 only: allow to print the document to a representation from which a faithful digital copy of the PDF content could be generated. When this flag is NOT specified (and PDF_ALLOW_PRINT is used), printing is limited to a lowlevel representation of the appearance, possibly of degraded quality. Acrobat viewers implement this limited mode of printing as “Print As Image”, except on UNIX systems, where this feature is not available.</td>
</tr>
<tr>
<td>PDF_ALLOW_ALL</td>
<td>3900</td>
<td>PDFAllowAll</td>
<td>no protection, all flags are raised</td>
</tr>
</tbody>
</table>

**Default:**

**PDF_ALLOW_NONE**
33.21 SetBookmarkDestination

[VPE Professional Edition and above]

Sets the destination for subsequently added bookmarks.

**method void VPE.SetBookmarkDestination(****

    BookmarkDestination [integer]  BookmarkDestination,
    long  Left,
    long  Top,
    long  Right,
    long  Bottom,
    double  ZoomFactor

)

*BookmarkDestination [integer]  BookmarkDestination*

**Bookmark Type, possible values are:**

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VBOOKMARK_DEST_NONE</td>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>No Destination</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VBOOKMARK_DEST_LTZ</td>
<td>1</td>
<td>LeftTopZoom</td>
</tr>
</tbody>
</table>
| Normal Destination (left, top, zoom are used) - this is the default
Display the target page with the coordinates (left, top) positioned at the top-left corner of the window and the contents of the page magnified by the zoom_factor. A null value for any of the parameters left, top, or zoom_factor specifies that the current value of that parameter is to be retained unchanged. |
| VBOOKMARK_DEST_FIT | 2    | Fit             |
| Fit the page in the window (no coordinates used)
Display the target page with its contents magnified just enough to fit the entire page within the window both horizontally and vertically. If the required horizontal and vertical magnification factors are different, use the smaller of the two, centering the page within the window in the other dimension. |
| VBOOKMARK_DEST_FITH | 3    | FitHorizontally |
| Fit the page in the window horizontally (top coordinate used)
Display the target page with the vertical coordinate top positioned at the top edge of the window and the contents of the page magnified just enough to fit the entire width of the page within the window. |
| VBOOKMARK_DEST_FITV | 4    | FitVertically   |
| Fit the page in the specified rectangle (all coordinates used)
Display the target page with its contents magnified just enough to fit the rectangle specified by the coordinates left, bottom, right and top entirely within the window both horizontally and vertically. If the required horizontal and vertical magnification factors are different, use the smaller of the two, centering the rectangle within the window in the other dimension. |
| VBOOKMARK_DEST_FITR | 5    | FitRectangle    |
| Fit the page in the specified rectangle (all coordinates used) |
Display the target page with its contents magnified just enough to fit the rectangle specified by the coordinates \textit{left}, \textit{bottom}, \textit{right} and \textit{top} entirely within the window both horizontally and vertically. If the required horizontal and vertical magnification factors are different, use the smaller of the two, centering the rectangle within the window in the other dimension.

\begin{verbatim}
VBOOKMARK_DEST_FITB  6        FitBounding
Fit the page's bounding rectangle in the window
Display the target page with its contents magnified just enough to fit its bounding box entirely within the window both horizontally and vertically. If the required horizontal and vertical magnification factors are different, use the smaller of the two, centering the bounding box within the window in the other dimension.

VBOOKMARK_DEST_FITBH  7        FitBoundingHorizontally
Fit the page's bounding rectangle horizontally in the window (top coordinate used)
Display the target page with the vertical coordinate \textit{top} positioned at the top edge of the window and the contents of the page magnified just enough to fit the entire width of its bounding box within the window.

VBOOKMARK_DEST_FITBV  8        FitBoundingVertically
Fit the page's bounding rectangle vertically in the window (left coordinate used)
Display the target page with the horizontal coordinate \textit{left} positioned at the left edge of the window and the contents of the page magnified just enough to fit the entire height of its bounding box within the window.
\end{verbatim}

\textbf{long Left, long Top, long Right, long Bottom}
position and dimensions

\textbf{double ZoomFactor}
zoom factor, e.g.: 0.5 \(= 50\%\); 1.0 \(= 100\%\); 2.0 \(= 200\%\); etc.

\textbf{Default:}

VBOOKMARK_DEST_LTZ
33.22 BookmarkStyle
[VPE Professional Edition and above]
Sets / returns the current bookmark style.

property BookmarkStyle [integer] VPE.BookmarkStyle

read / write; runtime only

Possible values:
any combination of the following flags, the values can be combined by adding there values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBOOKMARK_STYLE_NONE</td>
<td>0</td>
<td>None</td>
<td>[default]</td>
</tr>
<tr>
<td>VBOOKMARK_STYLE_BOLD</td>
<td>1</td>
<td>Bold</td>
<td>PDF 1.4 or higher</td>
</tr>
<tr>
<td>VBOOKMARK_STYLE_ITALIC</td>
<td>2</td>
<td>Italic</td>
<td>PDF 1.4 or higher</td>
</tr>
<tr>
<td>VBOOKMARK_STYLE_OPEN</td>
<td>4</td>
<td>Open</td>
<td></td>
</tr>
</tbody>
</table>

Default:
VBOOKMARK_STYLE_NONE

Example:

ActiveX / VCL:

// Bold and Italic styles will only work with PDF 1.4, activate it:
VPE.PDFVersion = VPE_PDF_ACROBAT_5

// Change the bookmark style to bold and italic:
VPE.BookmarkStyle = VBOOKMARK_STYLE_BOLD + VBOOKMARK_STYLE_ITALIC

.NET:

// Bold and Italic styles will only work with PDF 1.4, activate it:
VPE.PDFVersion = PDFVersion.Acrobat5

// Change the bookmark style to bold and italic:
VPE.BookmarkStyle = BookmarkStyle.Bold + BookmarkStyle.Italic
33.23 BookmarkColor

[VPE Professional Edition and above]

Sets the current bookmark color. Bookmark colors are supported by Acrobat Reader 5.0 (PDF v1.4) or higher.

**property Color VPE.BookmarkColor**

- read / write; runtime only

**Possible values:**
- any of the "COLOR_xyz" constants described in Programmer's Manual
- or **ActiveX / VCL**: any RGB value
- or **.NET**: any member of the .NET Color structure

**Default:**
- COLOR_BLACK
33.24 AddBookmark

[VPE Professional Edition and above]

Adds a bookmark to the document: a PDF document may optionally display a document outline on the screen, allowing the user to navigate interactively from one part of the document to another. The outline consists of a tree-structured hierarchy of bookmark items, which serve as a "visual table of contents" to display the document's structure to the user. Bookmarks are displayed in the left tree-view of Acrobat Reader.

The bookmark will have the currently assigned destination, style and color (see SetBookmarkDestination(), BookmarkStyle, BookmarkColor).

The target page for the destination is always the currently active page of the VPE document as you call this method.

In order to create a hierarchical tree structure, you can supply a parent bookmark as *parent*, i.e. the added bookmark will be a child of the supplied parent bookmark. To add a bookmark to the top-level of the hierarchy, specify a value of null for the *parent*.

The AddBookmark method returns a handle to the newly created bookmark, which can be used as *parent* for subsequent calls.

```vpe
method TVPEBookmark [pointer] VPE.AddBookmark(
    TVPEBookmark [pointer] Parent,
    string Title
)
```

*TVPEBookmark [pointer] Parent*
the parent bookmark or null for a top-level bookmark

*string Title*
the title of the bookmark

**Returns:**
the newly created bookmark, which can be used as *parent* for subsequent calls

**Remarks:**
Bookmarks are not shown within VPE documents. They are stored internally and are exported to PDF documents only.

Bookmarks are not stored within VPE document files. If you write a VPE document that contains bookmarks to a file, read the file later into memory and export it to PDF, the bookmarks are omitted. For the same reason, bookmarks will be missing in exported PDF files, if the VPE source document is created stream-based by using SwapFileName.

**Example:**

```vcl
// Bold and Italic styles will work only with PDF 1.4, activate it:
VPE.PDFVersion = VPE_PDF_ACROBAT_5

// Output some text on the current page:
VPE.VpePrint(1, 1, "Introduction")

// Set the style for newly added bookmarks to bold, italic, open:
VPE.BookmarkStyle = VBOOKMARK_STYLE_BOLD + VBOOKMARK_STYLE_ITALIC + VBOOKMARK_STYLE_OPEN
```
// Set the destination type for newly added bookmarks:
VPE.SetBookmarkDestination(VBOOKMARK_DEST_FIT, 0, 0, 0, 0, 1)

// Add a new bookmark to the top-level of the hierarchy:
Pointer ParentBookmark = VPE.AddBookmark(0, "Introduction")

// Append a new page to the VPE document:
VPE.PageBreak();

// Output some text on the current page:
VPE.VpePrint(1, 1, "Bookmarks explained")

// Add a new bookmark as child of the previously inserted bookmark.
// It will have the currently adjusted style, color and destination:
VPE.AddBookmark(ParentBookmark, "Bookmarks explained")

ActiveX, .NET, Java, ...:

// Bold and Italic styles will work only with PDF 1.4, activate it:
VPE.PDFVersion = PDFVersion.Acrobat5

// Output some text on the current page:
VPE.Print(1, 1, "Introduction")

// VB .NET
// Set the style for newly added bookmarks to bold, italic, open:
VPE.BookmarkStyle = BookmarkStyle.Bold + BookmarkStyle.Italic +
       BookmarkStyle.Open

// C# (must use | operator)
// Set the style for newly added bookmarks to bold, italic, open:
VPE.BookmarkStyle = BookmarkStyle.Bold | BookmarkStyle.Italic |
       BookmarkStyle.Open

// Set the destination type for newly added bookmarks:
VPE.SetBookmarkDestination(BookmarkDestination.Fit, 0, 0, 0, 0, 1)

// Add a new bookmark to the top-level of the hierarchy:
// (for Visual Basic .NET use "Nothing" instead of "null")
TVPEBookmark ParentBookmark;
ParentBookmark = VPE.AddBookmark(null, "Introduction")

// Append a new page to the VPE document:
VPE.PageBreak();

// Output some text on the current page:
VPE.Print(1, 1, "Bookmarks explained")

// Add a new bookmark as child of the previously inserted bookmark.
// It will have the currently adjusted style, color and destination:
VPE.AddBookmark(ParentBookmark, "Bookmarks explained")

For Visual Basic 6.0 use “Nothing” instead of “null” in the call to AddBookmark().
33.25 PDFALevel

[VPE Professional Edition and above]

VPE supports the export to the PDF/A format for long-term archival, according to ISO standard ISO 19005-1:2005, PDF/A-1b.

property PDFALevel [long] VPE. PDFALevel

write; runtime only

Possible values:

<table>
<thead>
<tr>
<th>ActiveX / VCL</th>
<th>Value</th>
<th>Enum</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPE_PDF_A_LEVEL_NONE</td>
<td>0</td>
<td>LevelNone</td>
<td>No PDF/A document is created. [default]</td>
</tr>
<tr>
<td>VPE_PDF_A_LEVEL_1B</td>
<td>1</td>
<td>Level1B</td>
<td>A PDF/A-1b document shall be created.</td>
</tr>
</tbody>
</table>

Default:

VPE_PDF_A_LEVEL_NONE

Remarks:

The PDF/A standard enforces several other document properties. When you set this property to VPE_PDF_A_LEVEL_1B, VPE sets the following properties automatically:

- PDFVersion = VPE_PDF_ACRONBAT_5
- FastWebView = false
- EmbedAllFonts = true
- Encryption = DOC_ENCRYPT_NONE

Furthermore you need to add a Color Profile to the PDF document, see the method AddColorProfile().

Example:

```pascal
Doc.PDFALevel = VPE_PDF_A_LEVEL_1B
Doc.AddColorProfile(
  "GTS_PDFA1",
  "sRGB_IEC61966-2-1",
  "Custom",
  "http://www.color.org",
  "sRGB_IEC61966-2-1",
  "sRGB_IEC61966-2-1")
Doc.WriteDoc("test.pdf")
```
33.26 AddColorProfile

[VPE Professional Edition and above]

Adds a Color Profile to the exported PDF document. A Color Profile is required when exporting to the PDF/A document format for long-term archival

```csharp
method void VPE.AddColorProfile(
    string Subtype,
    string OutputCondition,
    string OutputConditionIdentifier,
    string RegistryName,
    string Info,
    string FileName
)
```

string Subtype
the sub-type of the color profile. See the PDF specification for details.

string OutputCondition
the output condition for the color profile. Use "GTS_PDFA1", see the PDF specification for details.

string OutputConditionIdentifier
the output condition identifier of the color profile. Use "Custom", see the PDF specification for details.

string RegistryName
the registry name of the color profile. See the PDF specification for details.

string Info
the info dictionary entry of the color profile. See the PDF specification for details.

string FileName
the path and file name of the color profile file. VPE reads this file from hard disk end embeds it into the PDF document. VPE does not verify or convert the given file.

For convenience, VPE provides two built-in color profiles:
"sRGB_IEC61966-2-1" and "AdobeRGB1998"
If you specify either of both strings as FileName parameter, VPE will embed this profile from its internal memory. No external files are required.

Remarks:
VPE does not convert images to the given Color Profile. When adding images to a VPE document, you need that they match the Color Profile. Special care must be taken when using JPEG images: VPE embeds JPEG images in their original binary file format into PDF streams. That means, JPEG images in the CMYK format are embedded as CMYK into the document. Make sure they are converted to the given Color Profile, before importing them into a VPE document.
Example:

```csharp
Doc.PDFALevel = VPE_PDF_A_LEVEL_1B
Doc.AddColorProfile(
    "GTS_PDFA1",
    "sRGB_IEC61966-2-1",
    "Custom",
    "http://www.color.org",
    "sRGB_IEC61966-2-1",
    "sRGB_IEC61966-2-1")
Doc.WriteDoc("test.pdf")
```
33.27 ExtIntDA

Used internally for diagnostic tasks. **Do not use!**

```java
method long VPE.ExtIntDA(
    long i,
    long s
)
```
34 HTML Export

34.1 HTML Export

This section describes the methods and properties related to HTML Export.

A VPE Document is exported to the HTML file format by calling the method `WriteDoc()`.

The following properties are written as (meta-)tags to generated HTML documents:

- **Title** as `<title>` tag
- **Creator** as "generator" meta-tag
- **Author** as "author" meta-tag
- **Keywords** as "keywords" meta-tag
- **Subject** as "description" meta-tag

In addition, the settings for `DocExportPictureQuality` and `DocExportPictureResolution` are applied to converted bitmap images (i.e. where a source image is not in a browser-compatible format, for example TIFF, and is converted to PNG).

Objects which are exported as bitmaps – like barcodes, charts, polygons, etc. – are using the properties `DocExportPictureResolution`, `PictureExportColorDepth` and `PictureExportDither`.

We recommend to set `DocExportPictureResolution` to 96 DPI for HTML export, which is the resolution for screens. This also creates smaller images (regarding their size in bytes).

For important details about HTML Export and the features provided by VPE, please see the Programmer's Manual, chapter "The HTML Export Module".
34.2 HtmlScale

Sets the scaling for HTML output. It is recommended to set the scale below 1.0 so that the generated HTML documents are printable by browsers, which need page space for headers and footers they do insert.

**property double VPE. HtmlScale**

  read / write; runtime only

**Possible Values:**
  the scale of the generated HTML document, a scale below 1.0 shrinks the document, a scale above 1.0 expands the document

**Default:**
  0.75, which equals 75% of the original scale
34.3 HtmlWordSpacing

Sets the spacing between words for HTML output. If the specified HTML scale is too low, it can happen on some browsers that text will overflow on the right side of objects, especially on Firefox. In this case you can use this property to fix the problem.

**property double VPE. HtmlWordSpacing**

read / write; runtime only

**Possible Values:**

the word spacing of the generated HTML document, a spacing below 0 moves words closer together, a spacing above 0 expands the distance between words

**Default:**

0
34.4 HtmlRtfLineSpacing

When viewing exported documents which contain Rich Text in Mozilla browsers, the default line spacing is too large and text will overflow objects at the bottom. In this case set this property to true. This option improves appearance of RTF for Mozilla browsers, but worsens the result in Internet Explorer.

**property boolean VPE. HtmlRtfLineSpacing**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>shrink the line spacing for viewing with Mozilla browsers</td>
</tr>
<tr>
<td>False</td>
<td>do not change the line spacing</td>
</tr>
</tbody>
</table>

**Default:**

False
34.5 HtmlCopyImages

Specifies the mode for handling images of generated HTML documents.

If set to true, VPE copies / creates images in a destination directory named "<html file>.img". If an image is used multiple times, it is only stored once in the target directory. If the original images are of type PNG, JPG or GIF and the Rotation is zero degrees, the original image is copied. Otherwise a compressed PNG is created from the original.

If set to false, HTML links to the original locations of all images are written. In this case you may only use file types, which are supported by HTML browsers, i.e. PNG, JPG or GIF. Furthermore rotation is not applied. If you require rotated images, they should be stored on disk already in rotated form.

Windows platform: regardless of this property's value, VPE objects not supported by HTML are still exported as PNG files. If you don’t want this, do not use such objects.

property boolean VPE. HtmlCopyImages

read / write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>copy / create images</td>
</tr>
<tr>
<td>False</td>
<td>write HTML links to the original image locations</td>
</tr>
</tbody>
</table>

Default:
True

Remarks:
This property should be set to false, if exporting HTML to a memory stream.
34.6 HtmlPageBorders
If true, borders are added at page boundaries, i.e. a frame is drawn at the page boundaries.

**property boolean VPE. HtmlPageBorders**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>add borders at page boundaries</td>
</tr>
<tr>
<td>False</td>
<td>do not draw borders at page boundaries</td>
</tr>
</tbody>
</table>

**Default:**

False
34.7 HtmlPageSeparators
If true, pages are separated by horizontal rules.

property boolean VPE. HtmlPageSeparators
    read / write; runtime only

Possible Values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>separate pages by horizontal rules</td>
</tr>
<tr>
<td>False</td>
<td>do not separate pages by horizontal rules</td>
</tr>
</tbody>
</table>

Default: False
35 XML Export

35.1 XML Export

This section describes the methods and properties related to XML Export.
A VPE Document is exported to the XML file format by calling the method `WriteDoc()`.

The XML Document has a following hierarchic structure.

```xml
  VPE
  DOC
    FileDictionary
      Image
    Page
      VpeObjects (i.e. Line, PolyLine, Picture …)
```
### 35.2 XMLPictureExport

Specifies the mode for handling images of generated XML documents.

If set to true, VPE copies / creates images in destination directory named "<XML file>.img". If an image is used multiple times, it is only stored once in the target directory. If the original images are of type BMP, TIF, GIF, PCX, JPG, PNG and ICO, the original image is copied. Otherwise a compressed PNG is created from the original. The file path is written to the XML stream in the object block of the picture. The FileDictionary block of the XML stream is empty.

If set to false, pictures are encoded into Base-64. The Base-64 code will be written into the FileDictionary block of the XML stream with the relative path as the key. The encoded image will be embedded only once within the XML stream.

**property boolean VPE.XmlPictureExport**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Copy / create external images</td>
</tr>
<tr>
<td>False</td>
<td>Encode images into Base-64 and embed them within the XML stream.</td>
</tr>
</tbody>
</table>

**Default:**

False

**Remarks:**

This property should be set to false, if exporting XML to a memory stream.
36 ODT Export

36.1 ODT Export

This section describes the methods and properties related to ODT Export. OpenDocument Text is a free and open document standard, which is published as an ISO/IEC international standard, ISO/IEC 26300:2006 Open Document Format for Office Applications (OpenDocument) v1.0. Published ODF standards meet the common definitions of an open standard, meaning they are freely available and implementable.

ODT files can be imported by Microsoft Office 2003 and later through add-ons.

A VPE Document is exported to the ODT file format by calling the method WriteDoc().

The ODT Document is a zip-File, which includes a group of xml-files and picture files. The following example shows the structure of an exported ODT Document.

```
  v  ODT
     β  META-INF\manifest.xml
     β  Pictures\000000000001.png
     β  Meta.xml
     β  Settings.xml
     β  Content.xml
     β  Styles.xml
```

META-INF\manifest.xml

Manifest.xml lists the all files in this archives format again. If this file is missing, this document will not be represented correctly.

Folder “Pictures”

The used images are stored in the folder Pictures as copies.

Meta.xml

The brief information of the ODT Document are saved here. For example, the generator, the create time of this document, etc.
**Settings.xml**

The document-specific attitudes are saved in settings.xml.

**Styles.xml**

All used document formats are stored in styles.xml. For example, fonts, line styles, paragraph styles, and page layout, etc.

**Content.xml**

The file content.xml contains text contents of the document.

36.2 OdtTextWidthScale
Specifications the width scaling for text. It represents a value in percent. Normally, this value should be set to 1.0.
This is a global value, which applies to all exported text objects.

property double VPE.OdtTextWidthScale
read / write; runtime only

Possible Values:
The text width scale of the generated ODT document. If the value is below 1.0, characters are positioned more closely. When the scale is above 1.0, characters are expanded.

Default:
1.0, which equals 100% of the original text width
36.3 OdtTextPtSizeScale
Specifies the scaling of the point size for text.
This is a global value, which applies to all exported text objects.

property double VPE.OdtTextPtSizeScale
read / write; runtime only

Possible Values:
The scale of the text point size of the generated ODT document. If the value is below 1.0, text is shrunk. When the scale is above 1.0, text is expanded.

Default:
0.98, which equals 98% of the original text point size
36.4 OdtLineHeight

Specifies the scale of the text line height. It represents a value in percent. Normally, this value should be set to 1.0.

This is a global value, which applies to all exported text objects.

**property double VPE.OdtLineHeight**

read / write; runtime only

**Possible Values:**
The text line height of the generated ODT document. If the value is below 1.0, the distance between two lines of text is shrunk. If the scale is above 1.0, the distance between two lines of text is expanded.

**Default:**
1.0, which equals 100% of the original line distance.
36.5 OdtAutoTextboxHeight

Open Office and VPE render text differently. Therefore exported ODT documents do not look 100% identically to VPE documents. Sometimes a text box of Open Office is not capable of displaying all the text it contains, although VPE can. With this parameter you can specify, whether a text box in ODT is drawn with the same height as in VPE, or automatically with a compatible height. If this property is set to true, the text box height specified in VPE is used as the minimal height of the exported text box.

This is a global value, which applies to all exported text objects.

**property bool VPE.OdtTextboxHeight**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>Export text boxes with compatible height, so that all the text can be displayed.</td>
</tr>
<tr>
<td>False</td>
<td>Text boxes are exported with the same height as specified in VPE.</td>
</tr>
</tbody>
</table>

**Default:**

True
36.6 OdtPositionProtect

VPE can draw text boxes as high as a whole page. If such a text box is exported to an ODT document, the position of the text box might be moved by Open Office automatically. In order to protect the positions of exported text boxes, you can set this property to true.

This is a global value, which applies to all exported text objects.

**property bool VPE.OdtPositionProtect**

read / write; runtime only

**Possible Values:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>The position of exported text boxes is protected.</td>
</tr>
<tr>
<td>False</td>
<td>The position of exported text boxes is not protected.</td>
</tr>
</tbody>
</table>

**Default:**

True
37 Differences between the DLL and the ActiveX / VCL

37.1 Differences between the DLL and the Control

This section is for users that are working with the DLL and want to use the ActiveX / VCL, or vice versa.

Instead of properties and methods the DLL offers a set of functions. The events are the same (notification messages sent via "SendMessage()").

The DLL functions all start with the prefix "Vpe" and require a document handle. Since the ActiveX and VCL use the syntax "<Object> . <Method>" the document handle is not required and the prefix "Vpe" is left out.

Example:

**DLL:**
```c
VpeLine( hDoc, x, y, x2, y2 )
```

**Control:**
```c
<Object>.Line( x, y, x2, y2 )
```

(Visual Basic has problems with the keywords "Print, Write, Line and Scale". VB doesn't recognize that these are methods and properties of an ActiveX. So we implemented them twice: in VB use "VpePrint, VpeWrite, VpeLine and VpeScale").

All DLL functions, which just set or get a single value, are defined as properties in the ActiveX / VCL.

Example:

**DLL:**
```c
VpeSetPenSize( hDoc, 10 )
```

**Control:**
```c
<Object>.PenSize = 10
```

Example:

**DLL:**
```c
ps = VpeGetPenSize( hDoc )
```

**Control:**
```c
ps = <Object>.PenSize
```
A
A2 252, 294
A3 252, 294
A3Extra 252, 294
A3ExtraTransverse 252, 294
A3Rotated 252, 294
A3Transverse 252, 294
A4 252, 294
A4Extra 252, 294
A4Plus 252, 294
A4Rotated 252, 294
A4Small 252, 294
A4Transverse 252, 294
A5 252, 294
A5Extra 252, 294
A5Rotated 252, 294
A5Transverse 252, 294
A6 252, 294
A6Rotated 252, 294
ABORT 36, 66
Absolute 240
Acrobat4 826
Acrobat5 826
AddBookmark 850
AddColorProfile 853
AddColumn 641
AddGap 639
AddLegend 627
AddMailAttachment 516
AddMailReceiver 513
AddPolygonPoint 391
AddPolyPoint 372
AddRow 640
AddValue 626
AddXLabel 630
AddYLabel 632
AdobeDeflate 573
AfterAutoPageBreak Event 88
AfterAutoPageBreak Event - .NET 34
AfterAutoPageBreak Event - Java 62
AfterCloseWindow Event 82
AfterCloseWindow Event - .NET 28
AfterCloseWindow Event - Java 56
AfterControlChange 100
AfterControlChange - .NET 48
AfterControlChange - Java 77
AfterControlEnter 97
AfterControlEnter - .NET 45
AfterControlEnter - Java 74
AfterControlExit 99
AfterControlExit - .NET 47
AfterControlExit - Java 58
AfterDestroyWindow Event 80
AfterDestroyWindow Event - .NET 26
AfterDestroyWindow Event - Java 54
AfterFieldChange 101
AfterFieldChange - .NET 49
AfterFieldChange - Java 78
AfterMail Event 94
AfterMail Event - .NET 42
AfterMail Event - Java 72
AfterOpenFile Event 84
AfterOpenFile Event - .NET 30
AfterOpenFile Event - Java 58
AfterSaveFile Event 86
AfterSaveFile Event - .NET 32, 60
ALIGN_CENTER 405, 407, 465
ALIGN_JUSTIFIED 405, 407
ALIGN_JUSTIFIED_AB 405, 407
ALIGN_LEFT 405, 407, 465
ALIGN_RIGHT 405, 407, 465
AltdDividerNPosition 709
AltdDividerPenColor 711
AltdDividerPenSize 710
AltdDividerStyle 715
AltFirst 716
AltLast 716
Always 233
APlus 252, 294
Append 573, 576
AreaAbsolute 697
AsBoolean 770
Ascii 473, 575
AsInteger 768
AsString 767
AsJavaDateTime 774
AsNetDateTime 773
AsString 767
Author 827
Auto 156, 268, 269, 305, 382, 518, 682, 689, 783
AUTO_BREAK_FULL 291
AUTO_BREAK_NO_LIMITS 291
AUTO_BREAK_OFF 291
AUTO_BREAK_ON 291
AutoBreakMode 291
AutoDelete 203
AutoFontSize 716
Aztec 504, 790
AztecControl 500
AztecFlags 499
AztecID 503
AztecMenu 501
AztecMultipleSymbols 502
B
B2Of5 469
B2OF5Matrix 469
B4 252, 294
B4JisRotated 252, 294
B5 252, 294
B5Extra 252, 294
B5JisRotated 252, 294
B5Transverse 252, 294
B6Jis 252, 294
B6JisRotated 252, 294
BackwardDiagonal 386, 635
BadQuality 572
Band 225
Bar 697
Bar2DAlignment 472
Bar3D 697
Barcode 469, 790
Barcode Functions 461
Barcode Functions (2D) 471
BarcodeAddTextParms 464
BarcodeAlignment 465
BarcodeAutoChecksum 466
BarcodeMainTextParms 464
BarcodeThickBar 468
BarcodeThinBar 467
Base11 473
Base27 473
Base37 473
Base41 473
Bcc 513
BCP_BOTTOM 462, 463, 464
BCP_DEFAULT 462, 463, 464
BCP_HIDE 462, 463, 464
BCP_TOP 462, 463, 464
BCT_2OF5 469
BCT_2OF5MATRIX 469
BCT_CODABAR 469
BCT_CODE11 469
BCT_CODE128 469
BCT_CODE39 469
BCT_CODE39EXT 469
BCT_CODE93 469
BCT_CODE93EXT 469
BCT_EAN128 469
BCT_EAN13 469
BCT_EAN13_2 469
BCT_EAN13_5 469
BCT_EAN2 469
BCT_EAN5 469
BCT_EAN8 469
BCT_EAN8_2 469
BCT_EAN8_5 469
BCT_IDENTCODE 469
BCT_INTELLIGENT_MAIL 469
BCT_INTERLEAVED2OF5 469
BCT_ISBN 469
BCT_ISBN_5 469
BCT_LEITCODE 469
BCT_MSI 469
BCT_POSTNET 469
BCT_PZN 469
BCT_RM4SCC 469
BCT_TELEPENA 469
BCT_UPCA 469
BCT_UPCA_2 469
BCT_UPCA_5 469
BCT_UPCE 469
BCT_UPCE_2 469
BCT_UPCE_5 469
BeforeMail Event 93
BeforeMail Event - .NET 41
BeforeMail Event - Java 71
BeforeOpenFile Event 83
BeforeOpenFile Event - .NET 29
BeforeOpenFile Event - Java 57
BeforePrintNewPage Event 90
BeforePrintNewPage Event - .NET 38
BeforePrintNewPage Event - Java 66
BeforeSaveFile Event 85
BeforeSaveFile Event - .NET 31
BeforeSaveFile Event - Java 59
Begin 611
Binary 473
Bitmap 265
BkgColor 375
BkgGradientEndColor 377
BkgGradientMiddleColor 380
BkgGradientMiddleColorPosition 379
BkgGradientPrint 382
BkgGradientPrintSolidColor 384
BkgGradientRotation 381
BkgGradientStartColor 376
BkgGradientTriColor 378
BkgMode 374
BLACK 534
BLUE 534
BLUEGREEN 534
BmpExportOptions 574
Bold 848
BookmarkColor 849
BookmarkStyle 848
BorderStyle 172
BothAxis 675
Bottom 462, 463, 464, 472, 680
BottomLinePenColor 713
BottomLinePenSize 712
Box 389
BPlus 252, 294
BringPreviewToTop 146
BROWN 534
Build-In Paragraph Settings 536
Build-In Paragraph Settings: RTF Auto Page Break 548
BusyProgressBar 200

C
C# 7
Cancel 225
Cancelled 138
CanRead (.NET) 617
CanSeek (.NET) 618
CanWrite (.NET) 619
Caption 173
Cassette 268, 269, 305, 783
Cc 513
Ccittfax3 573
Ccittfax4 573
Center 405, 407, 465, 472
CenterH 472
CenterPreview 145
CenterV 472
Centimeter 202, 288
D
Danish 224
Dash 364, 367, 634
DashDot 364, 367, 634
DashDotDot 364, 367, 634
DataMatrix 479, 790
DataMatrixBorder 478
DataMatrixColumns 476
DataMatrixEccType 474
DataMatrixEncodingFormat 473
DataMatrixMirror 477
DataMatrixRows 475
DataSource Object - TVPDataSource 755
DataSourceCount 749
DataSourceObject 750, 777
DateTimeIsUTC 740
DbiJapanesePostcard 252, 294
DbiJapanesePostcardRotated 252, 294
Default 462, 463, 464, 483, 572, 573, 574, 575, 576
DefaultTabSize 543
DefineFooter 422
DefineHeader 421
DefineKey 219
Deflate 573
Delete 428
DeleteObject 332
Description 759, 776
DevCollate 278
DevColor 263
DevCopies 277
DevDuplex 264
DevEnum 248
DevEnumPaperBins 266
DevFileName 282
DevFromPage 279
Device 250
Device Control Properties 247
DevJobName 283
DevOrientation 251
DevPaperBin 269
DevPaperFormat 252
DevPaperHeight 259
DevPaperWidth 258
DevPhysPageHeight 276
DevPhysPageWidth 275
DevPrintableHeight 274
DevPrintableWidth 273
DevPrinterOffsetX 271
DevPrinterOffsetY 272
DevPrintQuality 261
DevScalePercent 260
DevSendData 284
DevToFile 281
DevToPage 280
DevTTOption 265
DevYResolution 262
DiagonalCross 386, 635
Differences between the DLL and the Control 875
DispatchAllMessages 154
Dither256 578
Dither256Wu 578
DitherMono 578
DitherNone 578
DivFirst 716
DividerPenColor 708
DividerPenSize 707
DividerStyle 714
DivLast 716
DKBLUE 534
DKGRAY 534
DKGREEN 534
DKORANGE 534
DKPURPLE 534
DKRED 534
DKYELLOW 534
DOC_COMPRESS_FLATE 157
DOC_COMPRESS_NONE 157
DOC_ENCRYPT_NONE 840
DOC_ENCRYPT_STREAM 840
DocContainsControls 810
DocData 790
DocExportPictureQuality 834
DocExportPictureResolution 833
DocExportType 156
DocFileReadOnly 170
DoHelp Event 87
DoObjectClicked Event 95
DoPrintDevData Event 92
Dot 364, 367, 634
DoUDOPaint Event 96
Download 265
Draft 261
Drawing Functions 363
DSheet 252, 294
DumpTemplate 720
DumpTemplatePage 721
Dutch 224

E
Ean128 469
Ean13 469
Ean13_2 469
Ean13_5 469
Ean2 469
Ean5 469
Ean8 469
Ean8_2 469
Ean8_5 469
Early Binding 18
Ecc000 474
Ecc010 474
Ecc040 474
Ecc050 474
Ecc060 474
Ecc070 474
Ecc080 474
Ecc090 474
Ecc100 474
Ecc110 474
Ecc120 474
Ecc130 474
Ecc140 474
Ecc200 474
Edition 231
EditProtection 134
Ellipse 393, 790
Email 225
E-Mail Functions 507
EmbedAllFonts 832
EnableClickEvents 554
EnableCloseButton 191
EnableMailButton 181
EnableMultiThreading 136
EnableURLs 137
encryption 840
Encryption 840
encryption key length 841
EncryptionKeyLength 841
End 66, 611
EngineRenderMode 289
English 224
Enhanced 3, 231
EnterPageno 225
Enterprise 3, 231
Envelope 268, 269, 305, 783
Envelope10 252, 294
Envelope11 252, 294
Envelope12 252, 294
Envelope14 252, 294
Envelope9 252, 294
EnvelopeB4 252, 294
EnvelopeB5 252, 294
EnvelopeB6 252, 294
EnvelopeC3 252, 294
EnvelopeC4 252, 294
EnvelopeC5 252, 294
EnvelopeC6 252, 294
EnvelopeC65 252, 294
EnvelopeDL 252, 294
EnvelopeInvite 252, 294
EnvelopeItalian 252, 294
EnvelopeManual 268, 269, 305, 783
EnvelopeMonarch 252, 294
EnvelopePersonal 252, 294
Error 225
ErrorOpen 225
ErrorWrite 225
Escape 219, 221
ESheet 252, 294
Events Generated by VPE -.NET Control 25
Events Generated by VPE - ActiveX 79
Events Generated by VPE - Java Control 51
Events Generated by VPE - VCL 103
Exceptions 4, 10, 16, 22
Executive 252, 294
ExportNonPrintableObject 325
Extended 311
ExternalWindow 171
ExtIntDA 855
Fast Web View 835
Field Object - TVPEField 762
FieldCount 760
FieldObject 761
FileAccess 138
FileCreate 138
FileVersion 138
FileName 758
FileOpen 138
FileRead 138
FileWrite 138
FindControl 817
FindFieldObject 751
FindVpeObject 754
Finnish 224
FirstIndent 537
FirstObject 334
FirstPage 225
Fit 846
FitBounding 846
FitBoundingHorizontally 846
FitBoundingVertically 846
FitHorizontally 846
FitRectangle 846
FitVertically 846
Fixed3D 172
FixedSingle 172
Flate 157
Flush (.NET) 616
Focus 152
Folio 252, 294
font subsetting 837
FontAscent 343
FontDescent 344
FontExternalLeading 348
FontInternalLeading 347
FontName 398
FontSize 399
FormField 704, 790
FormFieldControl 804
FormFieldFlags 716
FormField 703
ForwardDiagonal 386, 635
Frame 790
Free 214
French 224
Full 233, 291
FullPage 214, 219, 221, 225
G
German 224
GetCharacterHeight 345
GetDevEntry 249
GetDevPaperBinID 268
GetDevPaperBinName 267
GetFieldAsBoolean 738
GetFieldAsInteger 734
GetFieldAsJavaDate 746
GetFieldAsNetDate 744
GetFieldAsNumber 736
GetFieldAsOleDate 742
GetFieldAsString 732
GetFieldIsNull 728
GetFocusControl 816
GetPicturePageCount 446
GetPictureTypes 442
GetRangeHeight 432
GifExportOptions 576
GoodQuality 572
GotoPage 219, 221
Gradient 382
GradientEllipse 374
GradientLine 374
GradientRectangle 374
GradientRectangle 374
GRAY 534
GREEN 534
Grid 219, 221, 225
GridMode 184
GridVisible 185
GUILanguage 224
GUITheme 223

H
H 482
HasText 429
HatchColor 387
HatchStyle 386
Height 431
Help 219, 221, 225
HelpRequested Event - .NET 33
HelpRequested Event - Java 61
HIBLUE 534
Hide 462, 463, 464
High 261, 834
HIGREEN 534
HiQuality 572
Horizontal 264, 386, 635
hsBDiagonal 386, 651
hsCross 386, 651
hsDiagCross 386, 651
hsFDiagonal 386, 651
hsHorizontal 386, 651
hsNone 386, 651
hsVertical 386, 651
HTML 156, 518
HTML Export 857
HtmlCopyImages 861
HtmlPageBorders 862
HtmlPageSeparators 863
HtmlRtfLineSpacing 860
HtmlScale 858
HtmlWordSpacing 859
hWndPreview 227

I
Identcode 469
Important Note for Users of Visual Basic 14
In VPE, Charts internally consist of two basic parts 623
InBackground 184
INCH 202, 288

Info 219, 221
InForeground 184
Information 225
InsertAtBottomZOrder 331
InsertedVpeObject 800
InsertedVpeObjectCount 799
InsertedVpeObjectPageNo 801
InsertPage 329
Installed 509
Integer X in [0, 100] 572
IntelligentMail 469
Interaction 812
Interactive 3, 231
Interactive Objects 803
Interleaved2Of5 469
Introduction 1
Isbn 469
Isbn_5 469
IsEof 608
IsEof (ActiveX / VCL) 594
IsMAPInstalled 509
IsNull 764
ISO-8859-1 402
ISO-8859-10 402
ISO-8859-11 402
ISO-8859-13 402
ISO-8859-14 402
ISO-8859-15 402
ISO-8859-2 402
ISO-8859-3 402
ISO-8859-4 402
ISO-8859-5 402
ISO-8859-6 402
ISO-8859-7 402
ISO-8859-8 402
ISO-8859-9 402
IsoB4 252, 294
IsOpen 129
IsPreviewVisible 149
IsPrinting 246
Italian 224
Italic 848

J
JapaneseEnvelopeChou3 252, 294
JapaneseEnvelopeChou3Rotated 252, 294
JapaneseEnvelopeChou4 252, 294
JapaneseEnvelopeChou4Rotated 252, 294
JapaneseEnvelopeKaku2 252, 294
JapaneseEnvelopeKaku2Rotated 252, 294
JapaneseEnvelopeKaku3 252, 294
JapaneseEnvelopeKaku3Rotated 252, 294
JapaneseEnvelopeYou4 252, 294
JapaneseEnvelopeYou4Rotated 252, 294
JapanesePostcard 252, 294
JapanesePostcardRotated 252, 294
Jpeg 573
JpegExportOptions 572
JumpTop 147
Justified 405, 407
JustifiedAb 405, 407
K
Kanji 483
KeepLines 549
KeepNextParagraph 550
Keyboard 225
Keywords 830
Kind 790

L
L 482
Landscape 251, 303, 782
LargeCapacity 268, 269, 305, 783
LargeFormat 268, 269, 305, 783
LastError 138
LastInsertedObject 333
LastPage 225
Layout Functions 287
Ledger 252, 294
Left 405, 407, 465, 472, 680
LeftBottom 680
LeftIndent 538
LeftTop 680
LeftTopZoom 846
Legal 252, 294
LegalExtra 252, 294
Legend 684
Leitcode 469
Length 607
Letter 252, 294
LetterExtra 252, 294
LetterExtraTransverse 252, 294
LetterPlus 252, 294
LetterRotated 252, 294
LetterSmall 252, 294
LetterTransverse 252, 294
Level1B 852
LevelNone 852
License 135
Line 369, 697, 790
Linearized PDF 835
LineCount 433
Live Demo on our Websites 17
LoadTemplate 718
LoadTemplateAuthKey 719
LoQuality 572
Low 261
Lower 268, 269, 305, 783
LTBLUE 534
LTGRAY 534
LTGREEN 534
LTLTBLUE 534
LTORANGE 534
LTRED 534
LTYELLOW 534
Lzw 573

M
M 482
Macintosh 402
MacRoman 402
MAGENTA 534
Mail 219, 221
MailAccessDenied 138
MailAmbiguousRecipient 138
MailAttachmentNotFound 138
MailAttachmentOpenFailure 138
MailAttachmentWriteFailure 138
MailAutoAttachDocType 518
MailBadRecipType 138
MailCreate 138
MailDiskFull 138
MailDoc 523
MailFailure 138
MailInsufficientMemory 138
MailInvalidEditFields 138
MailInvalidMessage 138
MailInvalidRecips 138
MailInvalidSession 138
MailLoadMapi 138
MailLogonFailure 138
MailMessageInUse 138
MailNetworkFailure 138
MailNoMessages 138
MailNotSupported 138
MailReserved 138
MailSender 512
MailSubject 520
MailText 521
MailTextTooLarge 138
MailTooManyFiles 138
MailTooManyRecipients 138
MailTooManySessions 138
MailTypeNotSupported 138
MailUnknownRecipient 138
MailUserAbort 138
MailWithDialog 522
Management 127
Manual 268, 269, 305, 783
MAPIType 511
Master 726
MaxiCode 487, 790
MaxiCodeEx 489
Maximum 638
MaxScale 212
MaxScalePercent 213
Medium 261
Memory 138
Memory Streams 600
Memory Streams (ActiveX / VCL) 587
MFC: Using the VPE ActiveX with the MFC without placing it in a Dialog Resource 19
Middle 268, 269, 305, 783
MidQuality 572
Minimum 637
MinScale 210
MinScalePercent 211
Mm10 288
ModBarcode 138
ModBarcode2D 138
ModChart 138
ModeChar 336
ModeWord 336
ModGraphImp 138
ModGraphProc 138
ModPDF 138
ModZlib 138
Monochrome 263
Mouse 225
Msi 469

N
n/a 572
Name 775, 793
nBottom 312, 796
nBottomMargin 317, 784
Never 233
NextObject 802
NextPage 225
nFree 316
nHeight 314
nLeft 796
nLeft, nTop, nRight, nBottom 312, 796
nLeftMargin 784
nLeftMargin, nTopMargin, nRightMargin, nBottomMargin 317, 784
NoAbortDialog 237
NoAutoPageDims 237
NoLimits 291
None 157, 172, 365, 635, 636, 675, 680, 682, 684, 689, 846, 848
NoPageBreak 339, 340, 341, 342, 355, 356, 357, 358, 359, 360, 361
NoPen 365
Normal 834, 840
Norwegian 224
Note 252, 294
NotInstalled 509, 511
nRenderHeight 338
nRenderWidth, nRenderHeight 338
nRight 312, 796
nRightMargin 317, 784
nTop 312, 796
nTopMargin 317, 784
nUDOBottom 570
nUDOLeft 570
nUDOLeft, nUDOTop, nUDORight, nUDOBottom 570
nUDORight 570
nUDOTop 570
Null 790
NullValueText 766
nWidth, nHeight 314

O
ObjectClicked Event - .NET 43
ObjectClicked Event - Java 73
ObjectHandle 725, 756, 763, 779, 789
ObjectID 555
ODT 156, 518
ODT Export 867
OdtAutoTextboxHeight 872
OdtLineHeight 871
OdtPositionProtect 873
OdtTextPtSizeScale 870
OdtLineWidthScale 869
Off 291
Office2000 223
Office2003 223
Ok 225
OLIVE 534
On 291
OnAfterControlChange 125
OnAfterControlEnter 122
OnAfterControlExit 124
OnAfterFieldChange 126
OnAfterMail Event 119
OnAfterOpenFile Event 108
OnAfterSaveFile Event 110
OnAutoPageBreak Event 112
OnBeforeMail Event 118
OnBeforeOpenFile Event 107
OnBeforeSaveFile Event 109
OnCloseWindow Event 106
OnDestroyWindow Event 104
OnFail 233
OnHelp Event 111
OnlyOne 268, 269, 305, 783
OnObjectClicked Event 120
OnPrintDevData Event 117
OnPrintNewPage Event 115
OnRequestClose Event 105
OnRequestControlExit 123
OnRequestPrint Event - VCL 114
OnUDOPaint Event 121
Open 219, 221, 225, 848
OpenDoc 128
OpenFileDialog 158
OpenFileName 160
OpenProgressDialog 197
ORANGE 534
Orientation 782
Orig 513
Owner Password 843
OwnerPassword 843

P
Packbits 573
PageBreak 290, 339, 340, 341, 342, 355, 356, 357, 358, 359, 360
PageCount 292, 752
PageFirst 219, 221
PageFormat 294
PageHeight 302, 781
PageLast 219, 221
PageLeft 219, 221
PageObject 753
PageOf 225
PageOrientation 303
PageRight 219, 221
PageScroller 193
PageScrollerTracking 194
PageWidth 214, 219, 221, 225, 301, 780
PaperBin 305, 783
PaperView 205
ParagraphControl 551
PDF 156, 518
Read (ActiveX / VCL) 591
ReadDoc 166
ReadDocPageRange 167
ReadDocStream 168
ReadDocStreamPageRange 169
ReadPrinterSetup 286
Ready 225
reaPercent 697
RED 534
Refresh 155
Relative 240
RemovePage 330
RemoveSet 311
RENDER_BREAK 339, 340, 341, 342, 355, 356, 357, 358, 359, 360
RENDER_NO_BREAK 339, 340, 341, 342, 355, 356, 357, 358, 359, 360, 361
RENDER_SKIP_BREAK 339, 340, 341, 342, 355, 356, 357, 358, 359, 360
RenderAztec 505
RenderBoxRTF 356
RenderBoxRTFFile 358
RenderBoxRTFStream 360
RenderDataMatrix 480
RenderFormField 361
Rendering 335
RenderMaxiCode 488
RenderMaxiCodeEx 491
RenderPDF417 497
RenderPicture 349
RenderPictureDIB 354
RenderPictureResID 352
RenderPictureResName 353
RenderPictureStream 351
RenderPrint 339
RenderPrintBox 340
RenderQRCode 486
RenderRTF 355
RenderRTFFile 357
RenderRTFStream 359
RenderTextBlock 436
RenderWrite 341
RenderWriteBox 342
RequestClose Event 81
RequestControlExit 98
RequestControlExit - .NET 46
RequestControlExit - Java 75
RequestPrint Event - .NET 36
RequestPrint Event - ActiveX 89
RequestPrint Event - Java 64
Reserved48 252, 294
Reserved49 252, 294
Reset 437
ResetFontControl 839
ResetParagraph 547
ResolvedText 795
ResolveName 513
RestorePos 321
Right 405, 407, 465, 472, 680
RightBottom 680
RightIndent 539
RightTop 680
Rle 574
Rm4Scc 469
Rotation 322
RouteHelp 188
RTF 790
RTF Functions 525
RtfBraces 138
RtfColorTable 138
RtfFontTable 138
RtfOverflow 138
Rulers 201
RulersMeasure 202
S
Save 219, 221, 225
SaveFileDialog 159
SaveFileName 161
Scale 206
ScaleMode 214
ScalePercent 209
ScrollBottom 219, 221
ScrollDown 219, 221
ScrollLeft 219, 221
ScrollPageDown 219, 221
ScrollPageLeft 219, 221
ScrollPageRight 219, 221
ScrollPageUp 219, 221
ScrollRight 219, 221
ScrollTop 219, 221
ScrollUp 219, 221
Seek (.NET) 611
Seek (ActiveX / VCL) 597
Seek (Java, PHP, etc.) 612
SeekEnd (ActiveX / VCL) 598
SeekEnd (Java, PHP, etc.) 613
SeekRel (ActiveX / VCL) 599
SeekRel (Java, PHP, etc.) 614
SelectFont 397
Sending Mail on 64-Bit Windows 508
SendKey 221
SetAsDateTime 771
SetBarcodeParms 462
SetBookmarkDestination 846
SetChartYAutoGridStep 679
SetColor 633
SetDefOutRect 318
SetFieldAsBoolean 739
SetFieldAsDateTime 741
SetFieldAsInteger 735
SetFieldAsJavaDateTime 747
SetFieldAsNetDateTime 745
SetFieldAsNumber 737
SetFieldAsOleDateTime 743
SetFieldAsString 733
SetFieldNullValueText 731
SetFieldToNull 729
SetFocus 150
SetFocusControl 815
SetFocusControlByName 814
SetFocusToFirstControl 813
SetFont 396
SetFontAttr 405
SetFontControl 838
Understanding the Data Types and Declarations
(Delphi) 21
Understanding the Data Types and Declarations
(Visual Basic) 14
UnitTransformation 288
Unsure 509
Untouched 268, 305, 783
UpcA 469
UpcA_2 469
UpcA_5 469
UpcE 469
UpcE_2 469
UpcE_5 469
Upper 268, 269, 305, 783
Usage 225
UsageKeyboard 225
UsageMouse 225
User 682, 689
user password 842
UserDefined 224, 252, 294
UserPassword 842
UseSet 310
UseTempFiles 836
UseTemplateMargins 722
UseTemplateSettings 723
Using the VPE ActiveX in a CFormView derived class 19

V
VBAR2D_ALIGN_BOTTOM 472
VBAR2D_ALIGN_CENTER 472
VBAR2D_ALIGN_CENTER_H 472
VBAR2D_ALIGN_CENTER_V 472
VBAR2D_ALIGN_LEFT 472
VBAR2D_ALIGN_RIGHT 472
VBAR2D_ALIGN_TOP 472
VBAR2D_DATA_MATRIX_ECC000 474
VBAR2D_DATA_MATRIX_ECC010 474
VBAR2D_DATA_MATRIX_ECC040 474
VBAR2D_DATA_MATRIX_ECC050 474
VBAR2D_DATA_MATRIX_ECC060 474
VBAR2D_DATA_MATRIX_ECC070 474
VBAR2D_DATA_MATRIX_ECC080 474
VBAR2D_DATA_MATRIX_ECC090 474
VBAR2D_DATA_MATRIX_ECC100 474
VBAR2D_DATA_MATRIX_ECC110 474
VBAR2D_DATA_MATRIX_ECC120 474
VBAR2D_DATA_MATRIX_ECC130 474
VBAR2D_DATA_MATRIX_ECC140 474
VBAR2D_DATA_MATRIX_ECC200 474
VBAR2D_DATA_MATRIX_ENC.ASCII 473
VBAR2D_DATA_MATRIX_ENC.BASE11 473
VBAR2D_DATA_MATRIX_ENC.BASE27 473
VBAR2D_DATA_MATRIX_ENC.BASE37 473
VBAR2D_DATA_MATRIX_ENC.BASE41 473
VBAR2D_DATA_MATRIX_ENC.BINARY 473
Vbar2DDataTooLong 138
Vbar2DDataTooShort 138
Vbar2DDataWrongLength 138
Vbar2DDataZeroLength 138
Vbar2DEdgeOverForced 138
Vbar2DFormatDataInvalid 138
Vbar2DFormatNotAllowed 138
Vbar2DFormatOutOfRange 138
Vbar2DFormatTooLong 138
Vbar2DInvalidBorder 138
Vbar2DInvalidData 138
Vbar2DInvalidEcc 138
Vbar2DInvalidEdge 138
Vbar2DSelfTestFailed 138
Vbar2DSquareAspectLarge 138
Vbar2DSquareAspectSmall 138
Vbar2DSquareEdgeTooLarge 138
Vbar2DSquareEdgeTooSmall 138
Vbar2DSquareEvenOddMatch 138
Vbar2DSquareTooLarge 138
Vbar2DSquareTooLong 138
Vbar2DSquareTooShort 138
Vbar2DSquareWrongLength 138
Vbar2DUndefinedId 138
VBIN_AUTO 268, 269, 305, 783
VBIN_CASSETTE 268, 269, 305, 783
VBIN_ENVELOPE 268, 269, 305, 783
VBIN_ENVMANUAL 268, 269, 305, 783
VBIN_LARGECAPACITY 268, 269, 305, 783
VBIN_LARGEFMT 268, 269, 305, 783
VBIN_LOWER 268, 269, 305, 783
VBIN_MANUAL 268, 269, 305, 783
VBIN_MIDDLE 268, 269, 305, 783
VBIN_ONLYONE 268, 269, 305, 783
VBIN_SMALLFMT 268, 269, 305, 783
VBIN_TRACTOR 268, 269, 305, 783
VBIN_UNTOUCHED 268, 305, 783
VBIN_UPPER 268, 269, 305, 783
VBKG_GRD_ELLIPSE 374
VBKG_GRD_LINE 374
VBKG_GRD_RECT 374
VBKG_SOLID 374, 674
VBKG_TRANSPARENT 374, 674
VBOOKMARK_DEST_FIT 846
VBOOKMARK_DEST_FITB 847
VBOOKMARK_DEST_FITBH 847
VBOOKMARK_DEST_FITBV 847
VBOOKMARK_DEST_FITR 846
VBOOKMARK_DEST_FITV 846
VBOOKMARK_DEST_LTZ 846
VBOOKMARK_DEST_NONE 846
VBOOKMARK_STYLE_BOLD 848
VBOOKMARK_STYLE_ITALIC 848
VBOOKMARK_STYLE_BOLD 848
VBOOKMARK_STYLE_NONE 848
VBOOKMARK_STYLE_OPEN 848
VBOTTOM 312
VBOTTOMMARGIN 317
VCHARSET_DEFAULT 402
VCHARSET_ARABIC 402
VCHARSET_CYRILLIC 402
VCHARSET_GREEK 402
VCHARSET_HEBREW 402
VCHARSET_LATIN_1 402
VCHARSET_LATIN_2 402
VCHARSET_LATIN_3 402
VCHARSET_LATIN_4 402
VCHARSET_LATIN_5 402
VCHARSET_LATIN_6 402
VCHARSET_LATIN_7 402
VCHARSET_LATIN_8 402
X
X 636
XAxis 675
XAxisTitle 628
Xlabels 684
XML 156, 518
XML Export 865
XMLPictureExport 866

Y
YAxis 675
YAxisTitle 629
YELLOW 534

Z
ZlibBuffer 138
ZlibData 138
ZlibStream 138
ZlibVersion 138
ZoomFactor 225
ZoomIn 216, 219, 221, 225
ZoomOut 217, 219, 221, 225
ZoomPreview 215
ZoomTool 214
z-order 331